BRL-TR-3065

(4)

TECHNICAL REPORT BRL-TR-3065

BRL

AD-A214 470

EXTENSION OF A MODEL OF LIQUID INJECTION IN A REGENERATIVE LIQUID PROPELLANT GUN BASED UPON COMPARISON WITH EXPERIMENTAL RESULTS

GLORIA P. WREN
WALTER F. MORRISON

DECEMBER 1989

U.S. ARMY LABORATORY COMMAND

BALLISTIC RESEARCH LABORATORY
ABERDEEN PROVING GROUND, MARYLAND

89 11 [illegible]

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188
1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE				
4. PERFORMING ORGANIZATION REPORT NUMBER(S) BRL-TR-3065		5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION US Army Ballistic Rsch Lab	6b. OFFICE SYMBOL *(If applicable)* SLCBR-IB-B	7a. NAME OF MONITORING ORGANIZATION		
6c. ADDRESS *(City, State, and ZIP Code)* Aberdeen Proving Ground, MD 21005-5066		7b. ADDRESS *(City, State, and ZIP Code)*		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER		
8c. ADDRESS *(City, State, and ZIP Code)*		10. SOURCE OF FUNDING NUMBERS		
		PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE *(Include Security Classification)*
EXTENSION OF A MODEL OF LIQUID INJECTION IN A REGENERATIVE LIQUID PROPELLANT GUN BASED UPON COMPARISON WITH EXPERIMENTAL RESULTS

12. PERSONAL AUTHOR(S)
Wren, Gloria P. and Morrison, Walter F.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Year, Month, Day)*	15. PAGE COUNT
TR	FROM ______ TO ______		

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Experimental data currently collected for the regenerative liquid propellant gun (RLPG) in the injection region include liquid and chamber pressures and piston position. Utilizing this data, it has been possible to extrapolate liquid velocity and values for the discharge coefficient. However, models of the RLPG currently in use have required the discharge coefficient to be an input with liquid velocity determined based on steady Bernoulli flow. In an effort to eliminate the discharge coefficient as an input, the authors have developed a formulation, presented in earlier reports, which couples the motion of the regenerative piston and velocity of the injected liquid which allows a prediction of the discharge coefficient. The objective of this paper is to determine the utility of the model in predicting direct and indirect experimental measurements in the

SEE CONTINUATION ON REVERSE SIDE

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT		21. ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED [✓ SAME AS RPT.] ☐ DTIC USERS		Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Gloria P. Wren	22b. TELEPHONE *(Include Area Code)* (301) 278-6199	22c. OFFICE SYMBOL SLCBR-IB-B

DD Form 1473, JUN 86 — Previous editions are obsolete. — SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

19. ABSTRACT (Con't)

RLPG. In the paper two sets of experimental data are examined, both 30-mm test fixtures differing primarily in the use of resistive forces on the piston and transducer block. The utility of the injection model to predict the experimentally measured motion of the regenerative piston and the derived values of the discharge coefficient is assessed. In general, the model is a good description of the liquid injection process in a regenerative liquid propellant gun.

TABLE OF CONTENTS

INTENTIONALLY LEFT BLANK.

LIST OF FIGURES

LIST OF FIGURES (CON'T)

ACKNOWLEDGMENTS

The authors would like to express appreciation to Mr. Cris Watson of the Ballistic Research Laboratory and to the General Electric Company, Tactical Systems Department, for providing the data used in this report.

INTENTIONALLY LEFT BLANK.

I. INTRODUCTION

The development of large caliber gun systems utilizing liquid propellants in place of conventional solid propellants has periodically been investigated in the United States since the late 1940's. A number of liquid propellant concepts have been studied, including bulk loaded and direct injection using both bipropellants and monopropellants. However, research since the mid 1970's has focused on the regenerative liquid propellant gun (RLPG) shown in Figure 1. The characteristic features are the differential piston area, the injection orifice and the propellant reservoir.

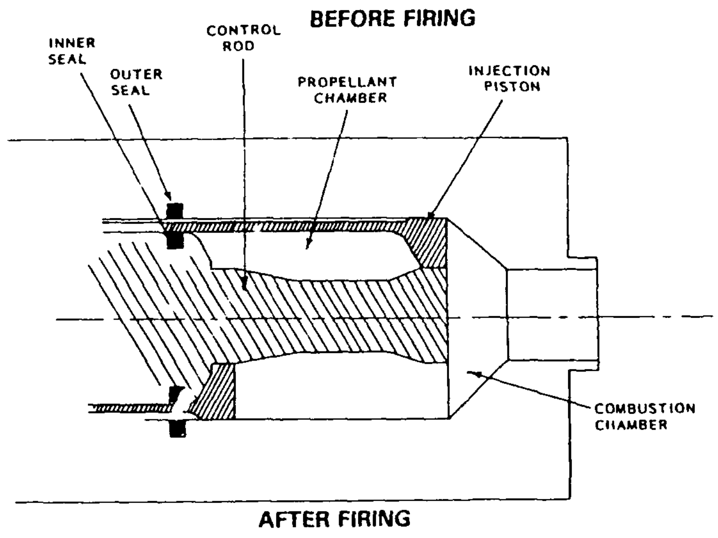

Figure 1. Regenerative Liquid Propellant Gun, Concept VI.

The interior ballistic process is initiated by firing an igniter which pressurizes the combustion chamber. The chamber pressure acting on the injection piston forces it to the rear, compressing the liquid in the reservoir. After an initial transient period, the pressure in the liquid reservoir will exceed the combustion chamber pressure as a result of the differential area across the injection piston. As the injection piston moves to the rear, opening the injection orifice, liquid propellant is injected into the combustion chamber, where it burns, accelerating the projectile.

The interior ballistic process in the regenerative liquid propellant gun is primarily controlled by the rate of injection of the liquid propellant, and, thus, by the motion of the regenerative piston. In the interior ballistic models developed to dar , the equation of motion for the regenerative piston has incorporated only the pressure and friction forces. Equations describing the injection of the liquid propellant from the liquid reservoir into the combustion chamber have generally employed a steady-state formulation with flow losses. These models, in general, neglect any direct coupling between the piston motion and liquid injection. In general, the acceleration of the liquid through the injector is also neglected, resulting in equations of the form,[1]

$$\dot{u}_p = \frac{1}{M_p}\left(P_3 A_p - \bar{P} A_R + A_3 \left[1 - \frac{A_3}{A_L} \right] \right)$$

$$v_3 = C_D \sqrt{2(\bar{P} - P_3)/\rho_L}$$

where C_D is a discharge coefficient adjusted to account for flow losses.

However, the above treatment has several inadequacies. Review of experimental data for the liquid propellant gun has suggested that the discharge coefficient has unexpectedly high values and is transient in nature in some experiments.[2,3] Thus, the formulation described above requires an empirical determination of the discharge coefficient for various nozzle configurations. Also, in the case of transient values, the discharge coefficient must be determined accurately over the critical start-up regime of the interior ballistic process. It is, therefore, of interest to develop a model which does not require a discharge coefficient, but which will accurately predict the motion of the regenerative piston and the liquid pressure history. Thus, a model has been suggested by the authors which is based on a time-dependent Bernoulli equation and on the extension of the control volume to include the entire propellant reservoir.[4-7]

The present work is a continuation of the treatment of the liquid injection process presented by Morrison and Wren which accounts for (1) the coupling between the regenerative piston motion and the injection of liquid propellant, and (2) the inertia of the liquid in the reservoir. In this paper, results of the simulation are presented and compared to experimental data. Two variations of the model are considered: one in which a simplified treatment of the pressure distribution in the liquid reservoir is utilized and one in which a complete treatment of the pressure distribution in the liquid reservoir is considered as part of the injection model. The distributions are derived from a modified Lagrange distribution with area change to account for the shape of the regenerative piston and the center bolt. In this paper the simple and full models are compared to each other and to experimental data.

Two sets of experimental data are examined, both from 30-mm test fixtures differing primarily in the use of resistive forces on the piston and transducer block and the geometry of the injection orifice. The utility of the simple version and the full version of the injection model in predicting the experimentally measured motion of the regenerative piston, the liquid pressure and the derived values of the discharge coefficient is assessed. In general, the simple model compares well with experimental data for Concept VIA fixtures, and the full model provides little additional modeling capability.

but which will accurately predict the motion of the regenerative piston and the liquid pressure history. Thus, a model has been suggested by the authors which is based on a time-dependent Bernoulli equation and on the extension of the control volume to include the entire propellant reservoir.[4-7]

The present work is a continuation of the treatment of the liquid injection process presented by Morrison and Wren which accounts for (1) the coupling between the regenerative piston motion and the injection of liquid propellant, and (2) the inertia of the liquid in the reservoir. In this paper, results of the simulation are presented and compared to experimental data. Two variations of the model are considered: one in which a simplified treatment of the pressure distribution in the liquid reservoir is utilized and one in which a complete treatment of the pressure distribution in the liquid reservoir is considered as part of the injection model. The distributions are derived from a modified Lagrange distribution with area change to account for the shape of the regenerative piston and the center bolt. In this paper the simple and full models are compared to each other and to experimental data.

Two sets of experimental data are examined, both from 30-mm test fixtures differing primarily in the use of resistive forces on the piston and transducer block and the geometry of the injection orifice. The utility of the simple version and the full version of the injection model in predicting the experimentally measured motion of the regenerative piston, the liquid pressure and the derived values of the discharge coefficient is assessed. In general, the simple model compares well with experimental data for Concept VIA fixtures, and the full model provides little additional modeling capability.

of the piston and the reservoir are approximated by straight line segments as indicated. The center bolt, which is fixed in these designs, is cast in the reference frame of the chamber. The origin of the coordinate system is fixed at the rear (left hand) end of the reservoir, and x is the coordinate along the bolt as shown in Figure 3. The piston moves rearward with a velocity u_p, and the points s_1, s_2, and s_3 are the coordinates of fixed stations on the inner contour of the piston with respect to the origin, as shown, such that these coordinates vary with time as the piston is displaced to the left. The right hand face of the control volume is attached to the chamber face of the piston, s_3, such that the control volume also varies with time.

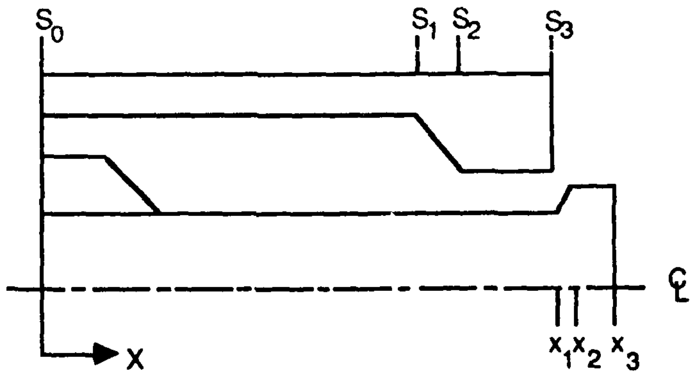

Figure 3. Control Volume for Concept VI

The derivation of both the simple and the full models has been presented in earlier reports and will not be repeated here. However, in general, the analysis begins with the one-dimensional momentum and continuity equations for the motion of the liquid in the reservoir and is written to include area change as the piston moves rearward. The area through which the fluid flows is a function of both time and position, since the contoured piston moves rearward over a contoured bolt. The equations of motion for the fluid (continuity and momentum) are then

$$\frac{\partial(\rho A)}{\partial t} + \frac{\partial(\rho v A)}{\partial x} = 0 \tag{1}$$

and,

$$\frac{\partial(\rho v A)}{\partial t} + \frac{\partial(\rho v^2 A)}{\partial x} = - A\frac{\partial P}{\partial x} \tag{2}$$

where ρ, v, A and P are all functions of both position and time.

The Lagrange assumption, density is a function of time only and is thus constant over the control volume such that the spatial derivative is zero, is a good approximation in the case of the LP reservoir since the liquid density only varies by about 4% over the entire ballistic cycle and the spatial variation over the reservoir at any given time is much less than this. Therefore, the Lagrange approximation, $\frac{\partial\rho}{\partial x} = 0$, is applied.

The analysis produces an unsteady Bernoulli equation and a relation between the exit pressure in the liquid at s_3 and the space-mean pressure in the liquid provided by the equation of state for liquid propellant. This formulation allows a coupling of the injection velocity of the liquid to the velocity of the piston by considering the momentum equation of the control volume including the regenerative piston. The momentum equation of the control volume shown in Figure 3, in the reference frame of the chamber, is

$$M_p \dot{\vec{u}}_p + \frac{\partial}{\partial t}\int_{CV} \vec{v}\rho dV + \int_{CS} \vec{v}\rho\vec{v}\; d\vec{A} = - \int P d\vec{A} \ , \tag{3}$$

where $d\vec{A}$ is the outward directed normal from the element of control surface.

Rewriting Equation (3),

$$M_p \dot{u}_p - \frac{\partial}{\partial t}\left\{\int_0^{s_3} \rho v A dx\right\} = P_3(A_p + A_3) - P_0 A_T + \rho v_3^2 A_3 \ , \qquad (4)$$

The unsteady Bernoulli equation and the force balance equation for the piston are the coupled, ordinary differential equations of motion governing liquid injection and injector piston motion. Experimental chamber pressure from an experimental gun firing provides a boundary condition for the problem.

In the full model the space-mean pressure is taken accurately to be

$$\overline{P}(t) = \frac{1}{V_R(t)}\int_0^{s_3} P(x,t)A(x,t)dx \qquad (5)$$

which treats the contours of the regenerative piston, while in the simple model the space mean pressure is considered only on the straight portion of the piston. This implies a much higher degree of complexity in the equations describing the liquid and piston accelerations in the full model compared to the simple model. The resulting system of ordinary differential equations in the full model is then:

$$\dot{v}_3 L_v^{eff}(t) - \dot{u}_p L_u^{eff}(t) = \frac{1}{\rho}[\overline{P}(t) - P_3(t)] - \frac{1}{2}v_3^2 + U^2(t) \qquad (6)$$

$$\dot{u}_p M_p^{eff}(t) - \dot{v}_3 m_L^{eff}(t) = P_3(t)[A_p + A_3] - \overline{P}(t)A_T + \rho v_3^2 A_3 - \rho[U^2 A(t)] \qquad (7)$$

$$\dot{x}_p = u_p$$

$$\dot{m}_L = -\rho_L A_H v_H$$

together with the equation of state for liquid propellant

$$\overline{P}(t) = P_l + \frac{K_1}{K_2}\left[\left(\frac{\rho_L}{\rho_0}\right)^{K_2} - 1\right] \tag{8}$$

where $\rho_L = \frac{m_L}{V_R}$ and where the terms required by the equations for the liquid and piston acceleration are defined in Appendix A.

Both the simple and full versions of the injection model are heavily geometry dependent, with the major differentiation between them being the detail captured in the development of the pressure gradient in the liquid reservoir. In the simple model the Lagrange pressure distribution with area change considers only the straight portion of the piston. That is, considering the interval $[0,s_1]$ as shown in Figure 3,

$$\overline{P}(t) = \frac{1}{l_{01}(t)}\int_0^{s_1} P(x,t)dx \tag{9}$$

the space-mean pressure is not treated over the entire reservoir and does not include the contour of the piston. This simplification, while not an accurate description of the pressure distribution in the liquid reservoir, was felt to be adequate for the resulting description of liquid injection and is a significant simplification of the model equations. The comparison with experimental data presented later in this report, in fact, shows that it is a good approximation. The resulting system of ordinary differential equations in the simple model replaces Equations (6) and (7) with

$$\dot{v}_3 + \dot{u}_p\left\{1 + \frac{1}{l_2}\left[l_1(t) + \frac{A_R l_1}{A_T}\ln\left(\frac{A_R}{A_3(t)}\right)\right]\right\} = \frac{1}{\rho l_1}[\bar{P}(t) - P_3(t)]$$

$$- \frac{1}{2}(v_3^2 + 2v_3u_p) - \rho(h_f + h'_f) \qquad (10)$$

$$\dot{u}_pM_p + \dot{v}_3\rho\left\{\frac{1}{2}l_1(t) + l_1 + l_2\right\}A_H = P_3(t)A_p - \bar{P}(t)A_T + \rho v_3^2 A_3 \qquad (11)$$

where

$$h'_f = \frac{1}{2}(v_3 + u_p)^2\left(\frac{1}{\psi} - 1\right)^2 \qquad (12)$$

is a representation of liquid loss through the orifice which has been adapted from pipe flow and h_f is friction acting on the piston. In the comparisons which follow both terms have been ignored. An assessment of possible friction acting on the piston was made, and in the fixtures examined these forces are very small compared to the forces associated with the liquid and chamber pressures.

III. *GEOMETRIC INTEGRALS FOR CONCEPT VIA GEOMETRY*

The geometric dependence of the area and volume terms, even in the simple model, requires that the geometric integrals be rederived for different relative placement of the points describing the piston and center bolt. That is, use is made in the model of area and volume relationships defined by the contours of the outer piston and center bolt. Since a derivation of the geometric integrals for the Concept VI pictured in Figure 1 was presented in an earlier paper,[4] the integrals derived here are those required in the simple model for the Concept VIA configuration depicted in Figure 2. It is possible to numerically evaluate the integrals. However, a numeric solution slows the computer simulation and was replaced, when possible, by an analytic solution.

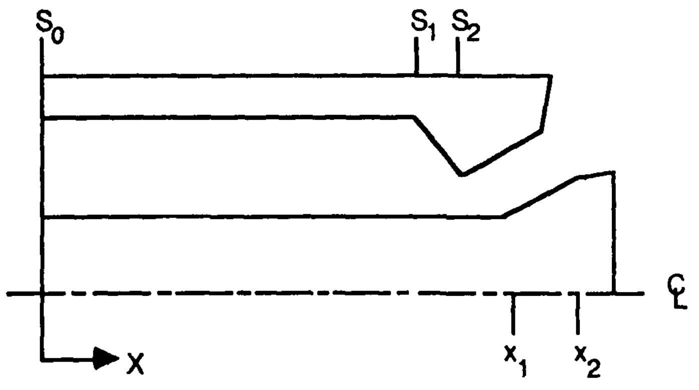

Figure 4. Control Volume for Concept VIA

Referring to Figure 4, the following terms are redefined for a Concept VIA fixture when utilizing the simple model. The radius of the piston is given by

$$R(x,t) = \left\{ R_1 + \frac{R_2 - R_1}{s_2 - s_1}(x - s_1)[1 - H(s_1 - x)] \right\} H(s_2 - x) + \left\{ R_2 + \frac{R_3 - R_2}{s_3 - s_2}(x - s_2)[1 - H(s_2 - x)] \right\} H(s_3 - x) \tag{13}$$

with R_i the radius at s_i. The radius of the bolt is given by

$$r_b(x,t) = r_1 + \frac{r_2 - r_1}{x_2 - x_1}(x - x_1)[1 - H(x_1 - x)] \tag{14}$$

with r_i the radius at x_i and where H(x) is the Heaviside function

$$H(x) = 0, x \leq 0$$
$$= 1, x > 0. \tag{15}$$

Then proceeding with a derivation similar to that described for the Concept VI fixture,[4] the geometric integrals are given by

$$L_{03}^{1}(t) = \frac{1}{2} l_{01}^{2}(t) \frac{A_L}{V_R} - \frac{\pi}{V_R M_1^2} C1 - \frac{\pi}{V_R M_2^2} C2 + \frac{1}{V_R} [A_L l_{01}(t) l_{13}(t) + V_{12} l_{23}] \tag{16}$$

$$L_{03}^{2}(t) = \frac{A_L}{V_R} \{ \frac{1}{2} l_{01}^{2}(t) + \left[\frac{A_L l_{01}(t)}{\pi M_1} + \frac{R_1 \left(\frac{1}{3} R_1^2 - r_b^2 \right)}{M_1^2} \right] C3 + C4$$
$$+ \left[\frac{A_L l_{01}(t) + V_{12}(t)}{\pi M_2} + \frac{R_2 \left(\frac{1}{3} R_2^2 - r_b^2 \right)}{M_2^2} \right] C5 + C6 \} \tag{17}$$

$$L_{03}^{3}(t) = l_{01}(t) + \frac{A_L}{\pi M_1} C3 + \frac{A_L}{\pi M_2} C5 \tag{18}$$

where l_{ij} represents the length from s_i to s_j and

$$C1 = R_1 \left(\frac{1}{3} R_1^2 - r_b^2 \right) (R_2 - R_1) + \frac{1}{2} (R_2^2 - R_1^2) r_b^2 - \frac{1}{12} (R_2^4 - R_1^4) \tag{19}$$

$$C2 = R_2 \left(\frac{1}{3} R_2^2 - r_b^2 \right) (R_3 - R_2) + \frac{1}{2} (R_3^2 - R_2^2) r_b^2 - \frac{1}{12} (R_3^4 - R_2^4) \tag{20}$$

$$C3 = \frac{1}{2 r_b} \ln \frac{(R_1 - r_b)(R_2 + r_b)}{(R_1 + r_b)(R_2 - r_b)} \tag{21}$$

$$C4 = \frac{1}{3}\frac{r_b^2}{M_1^2}\ln\frac{R_1^2 - r_b^2}{R_2^2 - r_b^2} - \frac{R_1^2 - R_2^2}{6M_1^2} \tag{22}$$

$$C5 = \frac{1}{2r_b}\ln\frac{(R_2 - r_b)(R_3 + r_b)}{(R_2 + r_b)(R_3 - r_b)} \tag{23}$$

$$C6 = \frac{1}{3}\frac{r_b^2}{M_2^2}\ln\frac{R_2^2 - r_b^2}{R_3^2 - r_b^2} - \frac{R_2^2 - R_3^2}{6M_2^2} \tag{24}$$

together with

$$M_1 = \frac{R_1 - R_2}{x_2 - x_1} \tag{25}$$

$$M_2 = \frac{R_2 - R_3}{x_3 - x_2}. \tag{26}$$

It is also necessary to find the derivatives of the geometric integrals given above. They are

$$\begin{aligned} \dot{L}^1_{03}(t) = {} & \frac{1}{V_R}[-u_p A_L l_{01}(t) + A_L l_{01}(t)\dot{l}_{12}(t) - A_L u_p l_{13}(t) + \dot{V}_{12} l_{23}] \\ & + \left[\frac{1}{V_R^2}u_p A_L\right]\left[\frac{A_L}{2}l_{01}^2(t) - \frac{\pi}{M_1^2}C1 - \frac{\pi}{M_2^2}C2 + A_L l_{01}(t) l_{13}(t) + V_{12} l_{23}\right] \end{aligned} \tag{27}$$

$$L_{03}^{2}(t)=\frac{1}{V_R}\left\{-u_pA_Ll_{01}(t)-\frac{u_pA_L^2}{\pi M_1}C3-\frac{u_pA_L^2}{\pi \dot{M}_2}+\frac{A_L}{\pi M_2}\dot{V}_{12}(t)\right\}$$

$$+\frac{u_pA_L}{V_R^2}\{\frac{1}{2}A_Ll_{01}^2(t)+\frac{A_L^2}{\pi M_1}l_{01}(t)C3+\frac{A_LR_1\left(\frac{1}{3}R_1^2-r_b^2\right)}{M_1^2}C3$$

$$+A_LC4+\frac{A_L^2}{\pi M_2}l_{01}(t)C5+\frac{A_L}{\pi M_2}V_{12}(t)C5$$

$$+\frac{A_LR_2\left(\frac{1}{3}R_2^2-r_b^2\right)}{M_2^2}C5+A_LC6\} \qquad (28)$$

$$\dot{L}_{03}^{3}(t)=-u_p \qquad (29)$$

Finally, it was decided to include a term which was not evaluated in the Concept VI simple model, the time rate of change of vent area. Since

$$\frac{dr}{dx}\bigg|_{A_3}=\frac{r_b(x_2)-r_b(x_1)}{x_2-x_1}=M_3 \qquad (30)$$

where M_3 represents the slope of the diverging section of the bolt which opens the vent area during the startup, and

$$A_3=\pi(R(s_3)^2-r_b^2(x)), \qquad x\in[x_1,x_2] \qquad (31)$$

then

$$\dot{A}_3(t)=2\pi u_pM_3, \qquad x\in[x_1,x_2]$$
$$=0 \qquad \text{otherwise}. \qquad (32)$$

The terms defined above were employed in the simple version of the injection model for Concept VIA and are compared to experimental data as shown in the following section.

IV. MODEL VALIDATION: 30-mm, Concept VIA

To assess the simple model, an experimental firing of a 30-mm, Concept VIA fixture with a damper and without Belleville springs, part of the General Electric variable volume series and labeled Shot 7, was utilized. The filtered, experimental data from this shot is shown in Figure 5. This fixture utilizes a damper on the outer piston primarily intended to slow the piston near completion of stroke. Also, absent from the fixture are the Belleville springs used to aid the startup of injection in earlier guns such as the 30-mm Concept VI fixture at BRL. The experimental data shows some initial unsteady motion in the piston with corresponding reflection in the slight liquid pressure oscillations, but overall smooth piston stroke. The unfiltered chamber pressure shows the presence of high frequency oscillations associated with much of the liquid propellant data. The shot is felt to be typical of 30-mm, Concept VIA data.[8]

As noted previously, the injection model currently uses experimental combustion chamber pressures as a boundary condition. However, the model does not require the input of a discharge coefficient for the liquid injection from the reservoir to the combustion chamber, predicting this value instead. The input consists of lengths, areas and volumes associated with the gun. Although it is possible to model the damper, the damper pressure was included as an additional boundary condition since the interest here is assessment of the injection model. The momentum equation for the control volume is then modified to

$$M_p \ddot{u}_p - \frac{\partial}{\partial t}\int_0^{s_3} \rho(t) v(x,t) A(x,t)\, dx = P_3(t)(A_p + A_3(t)) - P_0(t) A_V - P_D(t) A_D + \rho(t) v_3(t)^2(t) A_3(t) \qquad (33)$$

where P_D is the pressure of the damper and A_D is the projected area against the damper.

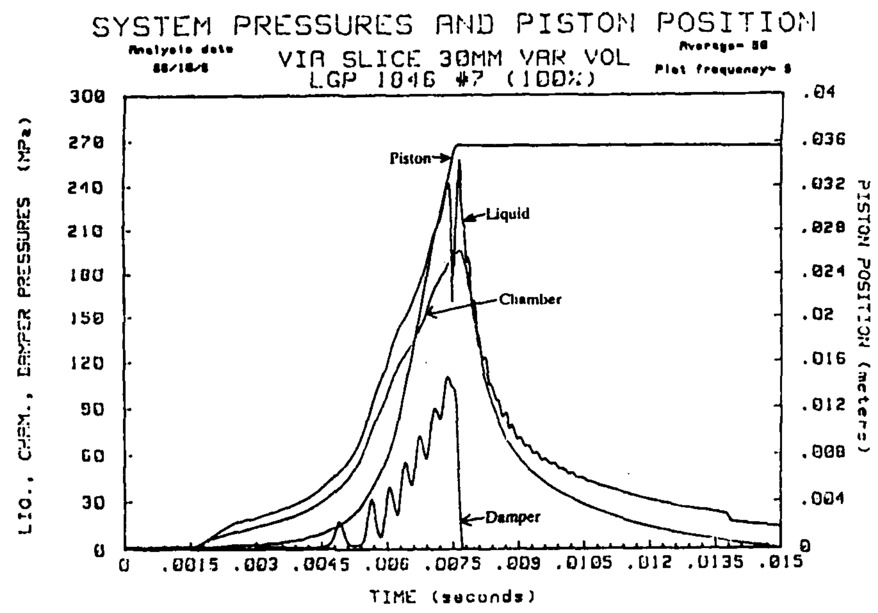

Figure 5. <u>Experimental Chamber Pressure, Liquid Pressure, Damper Pressure, and Piston Position from a 30-mm, Concept VIA RLPG (GE Shot 7).</u>

It is noted that the unfiltered experimental chamber pressure has high frequency oscillations. Thus, it is necessary to filter the experimental data before an acceptable boundary condition can be given. Since the frequencies in the combustion chamber have not yet been ascribed a specific physical significance, a reasonable filtered fit to the experimental data was sought, in this case a 5KHz low pass filter with a 100Hz transition. An overlay of the filtered and unfiltered data shows that the structure of the pressure-time curve has been maintained, but the oscillations have been removed. It is noted, however, that the specific filter utilized can affect the predicted values of the discharge coefficient.

The nonlinear ordinary differential equations describing liquid injection in the simple model together with the boundary conditions of chamber pressure and damper pressure are then numerically solved on an IBM-AT personal computer using SDRIV,[9] an efficient and robust computer code for the solution of initial value problems for ordinary differential equations which solves both stiff and non-stiff problems. The system of nonlinear ordinary differential equations posed in this paper was solved as a stiff problem. A listing of the input file can be found in Appendix B.

A comparison of the simulation to measured experimental piston position and liquid pressure is shown in Figures 6 and 7, respectively. The comparison of piston position shows good overall agreement both qualitatively and quantitatively, although several of the details in the experimental data are not captured. The model does not reflect the somewhat bumpy startup of the experimental piston, and it does not slow to match experimental data at the end of stroke. A comparison of liquid pressure shows similar good agreement although the model does not reflect the small oscillations in the experimental pressure. The computed discharge coefficient is shown in Figure 8. It exhibits a quick rise to steady state with a mean value of approximately 0.95, a value generally accepted for these fixtures. The value of the discharge coefficient derived from the experimental data is shown in Figure 9. The experimental values of the discharge coefficient in Figure 9 are not measured directly and are subject to a high degree of inaccuracy since a slight inaccuracy in calculating the vent area can significantly affect the derived values of the discharge coefficient. After a rather long transient period, the quasi-steady state value of the experimental discharge coefficient appears to be approximately 0.95. Therefore, considering the mean values, the experimental and simulated discharge coefficients are felt to be in reasonable agreement.

In general, the simple model appears to be an accurate representation of experimental piston position and liquid pressure with chamber and damper pressures included as boundary conditions for a Concept VIA, 30-mm RLPG. The model requires the geometry of the fixture as input, with the discharge coefficient predicted by the model. It appears that the more complex, full model is not required to describe the injection process in the Concept VIA fixture.

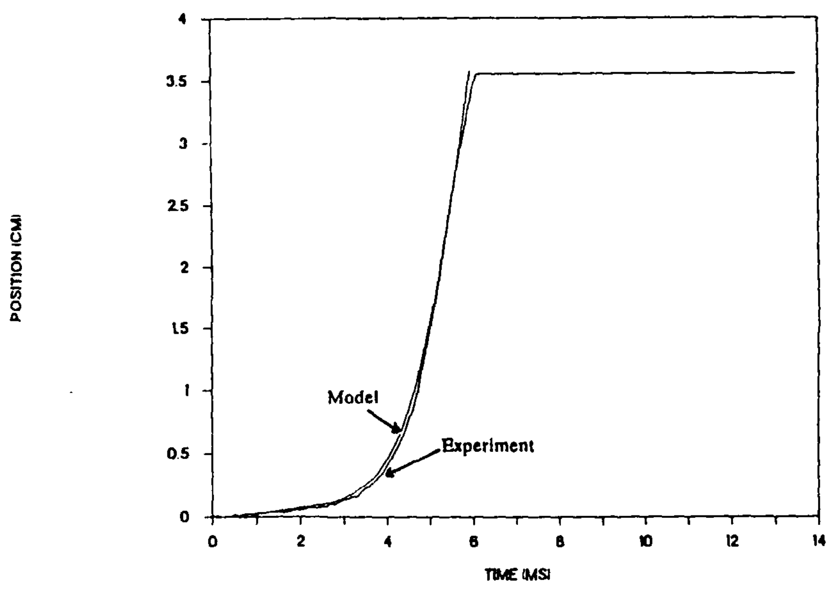

Figure 6. Comparison of Simple Injection Model with Experimental Piston Position from a 30-mm, Concept VIA RLPG (GE Shot 7).

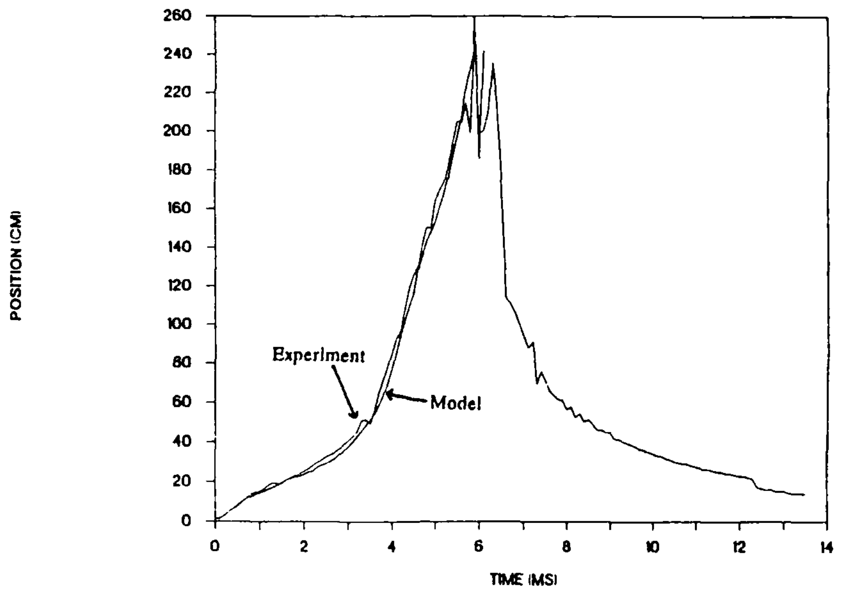

Figure 7. Comparison of Simple Injection Model with Experimental Liquid Pressure from a 30-mm, Concept VIA RLPG (GE Shot 7).

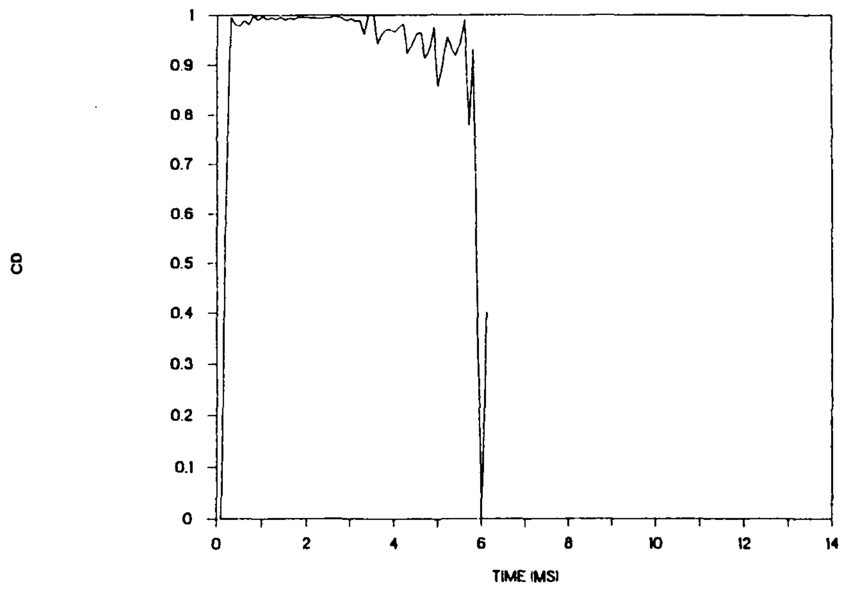

Figure 8. Predicted Values of the Discharge Coefficient from the Simple Injection Model for a 30-mm, Concept VIA RLPG (GE Shot 7).

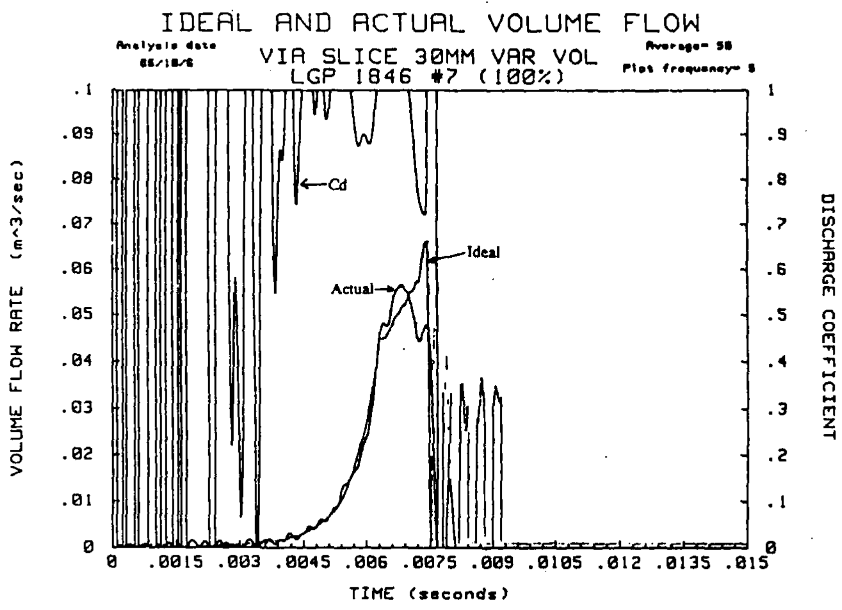

Figure 9. <u>Derived Values of the Discharge Coefficient from Experimental Pressures and Derived Ideal and Actual Volume Flow for a 30-mm, Concept VIA RLPG (GE Shot 7).</u>

V. MODEL APPLICATION: 30-mm, Concept VI

The original impetus for the injection model was reports of unexpectedly high values of the discharge coefficient.[2,3] The 30-mm, Concept VI RLPG at the Ballistic Research Laboratory became the focus of investigation, and it is a firing labeled Round 8 that has received the most scrutiny. In an effort to explore this data set, a number of approaches were taken. In this section an assessment is made of the simple model's ability to capture the experimental piston position and liquid pressure in Round 8. This required expanding the model by adding two additional simultaneous equations describing the motion of the transducer block against the Belleville springs. The required equations are

$$\ddot{y} = \frac{1}{M_T}[P_R(t)A_T - ky] \quad (34)$$

and

$$\dot{y} = \ddot{y} \quad (35)$$

where y is the position of the spring from its initial position at 0.0 and the spring constant, k, is determined experimentally from a measurement of the springs alone.

The experimental piston displacement, liquid pressure and filtered chamber pressure for a 30-mm, Concept VI RLPG (BRL Round 8) is shown in Figure 10. The piston begins to move at about 1.25 ms, travels approximately 0.5 cm, abruptly stops at about 3.5 ms, hesitates briefly and then again accelerates and smoothly completes its stroke. This interrupted piston motion is caused by the rear transducer block moving rearward against a set of Belleville springs in order to permit the piston to clear an O-ring seal on the forward end of the center bolt. Injection is designed to begin with compression of the Belleville springs and unseating of the O-ring. When the springs are fully compressed, the transducer block abruptly stops, as does the reservoir and the piston. The piston then accelerates rearward again as liquid injection begins and completes its stroke. The propellant in the reservoir is much stiffer than the combustion gases, and thus reflects the abrupt variations in the piston motion. As the Belleville springs begin to compress, a small oscillation in liquid pressure is observed at about 3.0 ms. When the transducer block suddenly stops at about 3.5 ms, the momentum of the piston is absorbed by the liquid, producing the relatively large pressure oscillations from 3.5 to 5.0 ms. Although initially undamped, as the injection area opens the oscillations are rapidly damped. Similarly, as the piston reaches the rear taper, which reduces the liquid injection area, the liquid pressure rises sharply as the piston is decelerated. The liquid gauge fails just as damping begins at about 8 ms.

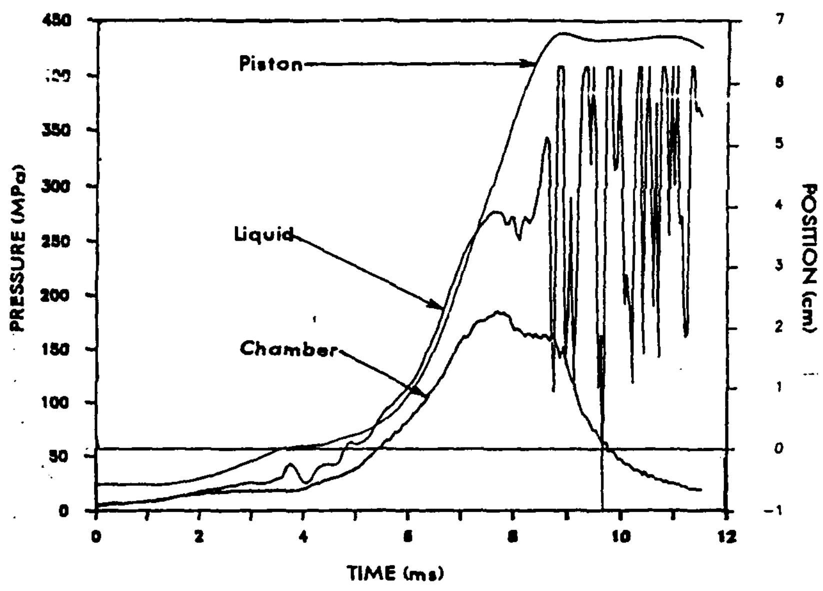

Figure 10. Experimental Chamber Pressure, Liquid Pressure and Piston Position for a 30-mm, Concept VI RLPG (BRL Round 8).

The simple model employing the Concept VI geometry with the inclusion of a Belleville spring model together with experimental combustion chamber pressure as a boundary condition was utilized as a model. The input file can be found in Appendix C. The comparison of the simulation with experimental liquid pressure and piston position is shown in Figures 11 and 12, respectively, where the zero in time has been chosen to coincide with the initial rise in combustion chamber pressure. The oscillation in liquid pressure at 2.4 ms and the corresponding flattening of the piston position is associated with the bottoming out of the Belleville springs. The major observation is that, although the model is in general agreement with experiment, the timing of events appears to be inaccurate. Although the piston velocity, as evidenced by the slope of the piston position versus time curve in figure 8, appears to agree with the experiment, the Belleville springs bottom out too quickly in the model as evidenced by the both the early flattening of the piston position curve and the early oscillations in the liquid pressure. Two possibilities were considered for this inconsistency. First, it was hypothesized that there may be friction associated with the seals and the grease column which is not captured by the model. Secondly, it is noted that several model assumptions are not accurate representations of the physical problem.

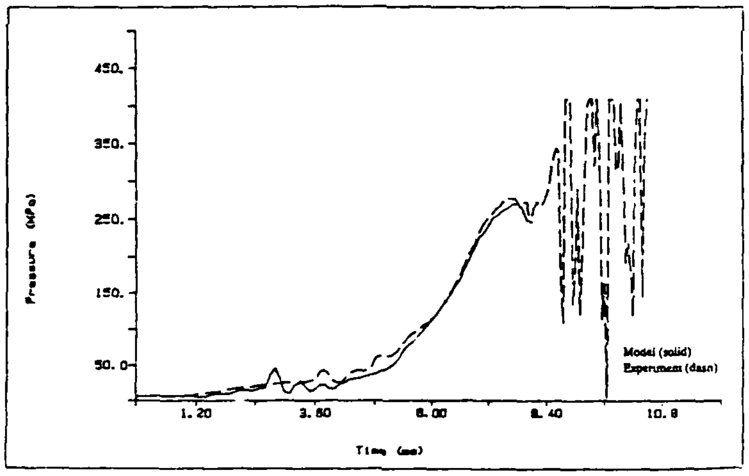

Figure 11. Comparison of Simple Model Simulation and Experimental Liquid Pressure for a 30-mm, Concept VI RLPG (BRL Round 8).

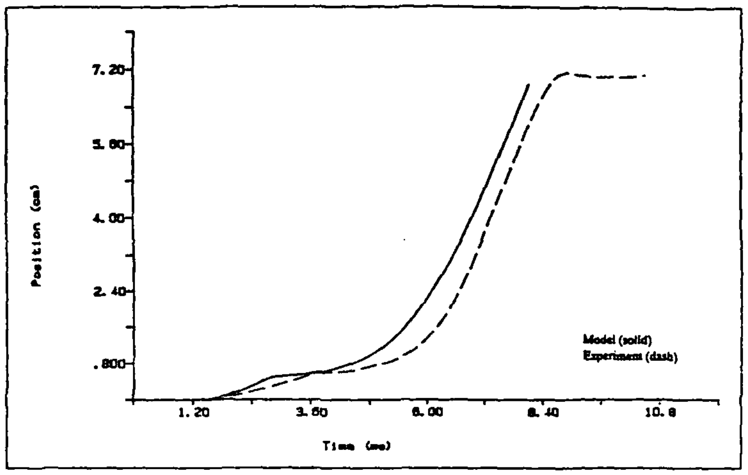

Figure 12. Comparison of Simple Model Simulation and Experimental Piston Position for a 30-mm, Concept VI RLPG (BRL Round 8).

To explore the possibility of friction from the seals and grease affecting piston motion during the start-up regime when the piston, liquid propellant and transducer block are moving rearward against the Belleville springs, a separate point mass-spring model was written. The point mass-spring model was utilized to compute the expected timing of the event of the bottoming of the Belleville springs. Until the bottoming of the springs, the liquid reservoir is sealed by an O-ring. Thus, during the initial motion, the transducer block mass, liquid propellant mass and piston mass are considered a point mass moving against the Belleville springs. The spring coefficients were provided by direct measurement of the springs alone. The model equation is

$$\ddot{x} = \frac{P_c(t)A_c}{M} - \omega^2 x \qquad with \qquad \omega^2 = \frac{k}{M} \tag{36}$$

where $P_c(t)$ is the combustion chamber pressure, A_c is the projected area of the liquid in the reservoir against the springs, k is the spring constant, and M is the point mass consisting of piston mass, liquid propellant mass and transducer mass. The liquid propellant in the liquid reservoir is initially prepressurized to 6.6 MPa which forces the piston forward against a crash ring and moves the transducer block rearward against the Belleville springs 0.177 cm. In this position, the system starts from rest and begins motion in response to the rise in combustion chamber pressure.

Experimental measurement of the Belleville springs shows that the full displacement of the springs is 0.422 cm. Since the liquid reservoir is filled with propellant, and the chamber pressures are low, the liquid propellant compressibility can be ignored during the first few milliseconds. Hence, the expectation is that, for the movement of the transducer block against the springs, the piston will move somewhat further. Since the piston and the transducer will not displace the same amount during the startup (the volume displaced by linear motion of the transducer is more than that displaced by the same movement of the piston), it is not possible to compare experimental piston displacement with the displacement of the transducer predicted by the model. The interest, then, is the timing of the event. As can be seen from Figure 10, the experimental liquid pressure reflects the bottoming of the springs at about 3.65 ms as evidenced by the sharp rise in liquid pressure. The point mass model predicts that the springs bottom out at 3.60 m. Thus, there is no evidence of significant friction affecting piston motion during the start-up regime. Therefore, the simple model's inability to capture the initial piston motion is not explained by friction in the system.

Secondly, several representations in the simple model do not accurately reflect the physical problem. The major concerns are addressed in the following section. However, for completeness, the following inconsistencies between the model and the physical problem in Round 8 are noted.

(1) The fixed zero in the control volume is displaced 0.177 cm in the physical problem by the pre-pressurization of the liquid reservoir. Thus, the lengths used in the model from the zero position are slightly inaccurate.

(2) It is assumed implicitly in the model that the transducer block and the piston move rearward as a unit during startup maintaining the lengths on $[0,s_1]$. To conserve the liquid reservoir volume, the piston will move slightly further than the springs due to the variation in area.

(3) The momentum equation in the model does not account for a moving rear boundary. In the physical problem the rear transducer block moves against the springs.

(4) Although the initial vent area in the experiment is zero and remains zero until the O-ring is expelled, the model will not allow a zero vent area since it occurs as a divisor. Thus, although the initial vent area in the simulation is kept as small as possible to allow the code to run, a small amount of liquid is expelled during the startup region.

(5) The pressure gradient in the liquid reservoir was considered over the straight portion of the piston, as a modeling simplification, instead of over the entire reservoir.

(6) Although the contour of the center bolt is reflected in the vent area, the geometry of the center bolt was not included in the evaluation of the area integrals.

Of the concerns listed above, only the simplification of the pressure gradient was expected to affect the solution. Thus, the most promising extension of the simple model appeared to be a precise statement of the space-mean pressure, that is, application of the full model. The extension of the pressure gradient to the entire reservoir significantly complicates the model, and it was necessary to both rederive the governing equations and to move the equations to a mainframe computer for solution. In the next section, the results from the full model with inclusion of the complete pressure gradient is examined.

In general, although the simple model is an adequate description of the 30-mm, Concept VIA RLPG, and captures the basic physics of the 30-mm Concept VI RLPG, it does not accurately model the detail associated with the early start-up regime in the Concept VI fixture.

VI. COMPARISON OF SIMPLE AND FULL INJECTION MODELS

To further explore the discrepancies between the simulation and the experimental data from Round 8 discussed in Section VI, the simple model was expanded to consider the full pressure distribution by removing the restriction of considering the space-mean pressure only on the straight portion of the piston in the derivation of the Lagrange pressure distribution with area change. As shown earlier in Appendix A, the resulting integrals are complex, and, although possible to analytically simplify, do require numeric integration to determine their final values. A listing of the code can be found in Appendix D. In this section the full and simple models are compared to each other and to the experimental data of Round 8.

As an intermediate step in the comparison, and to validate the analysis, the full Lagrange pressure and velocity distributions with area change in the liquid reservoir were compared to a one-dimensional simulation of the reservoir[10] since no experimental data exists for the liquid other than a single pressure measurement at each timestep. The full Lagrange pressure and velocity distributions have been derived elsewhere,[5] and are a result of considering the space-mean pressure through the entire liquid reservoir.

To assess the accuracy of the resulting model of pressure and velocity distribution in the liquid reservoir, the Lagrange model is compared with a one-dimensional simulation in Figures 13 and 14 at a mid-stroke position of the piston. As is evident from the figures, the models are in excellent qualitative and quantitative agreement with a difference of less than 1% in liquid propellant velocity and pressure at the exit of the liquid propellant from the liquid reservoir into the combustion chamber. Thus, the inclusion of the full pressure gradient into the injection model was felt to more accurately reflect the actual conditions in the reservoir.

Accordingly, the full model incorporates a consideration of the complete geometry of the piston. The contour of the center bolt was not included in the geometric integrals, but could be at the expense of more complication. In the figures which follow the simple and full injection models are compared to experimental data from the 30-mm, Concept VI RLPG firing labelled Round 8 just after the Belleville springs have bottomed out. Thus, the zero in time has been chosen to correspond to the bottoming of the springs at approximately 3.65 ms in the original experimental data. The initial conditions of piston position and velocity were determined from experimental data.

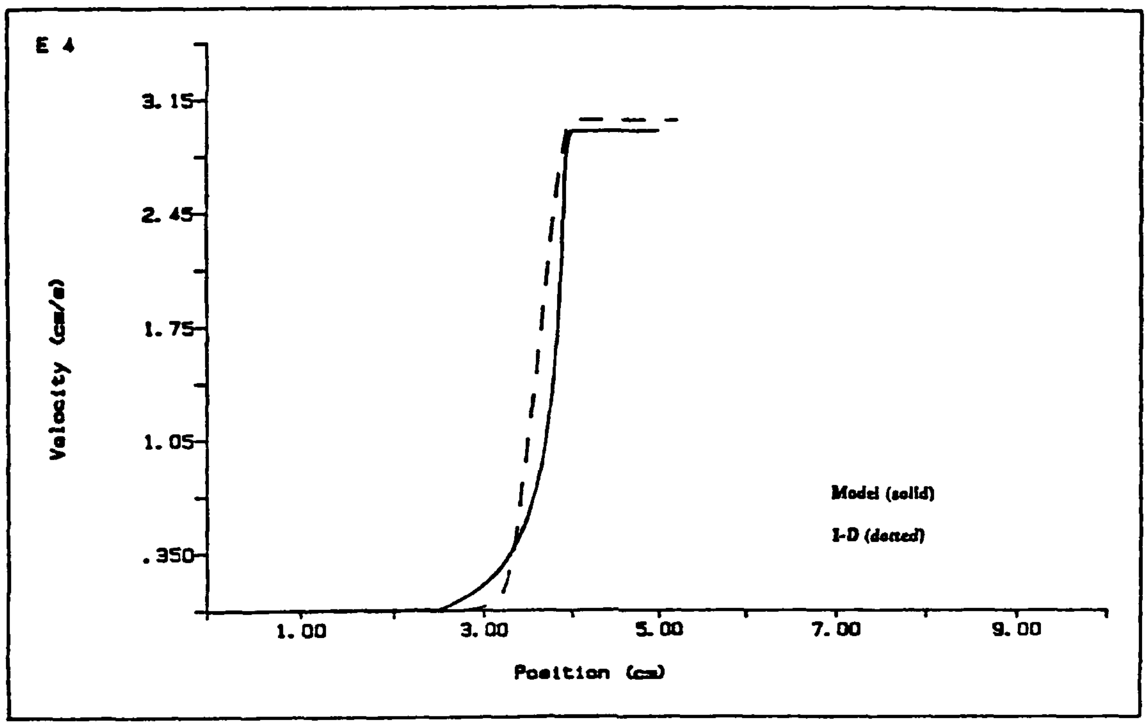

Figure 13. Comparison of Velocity Distributions from the Lagrange Pressure Distribution with Area Change Model (Solid Line) and a 1-Dimensional Simulation (Dotted Line) at Mid-Stroke.

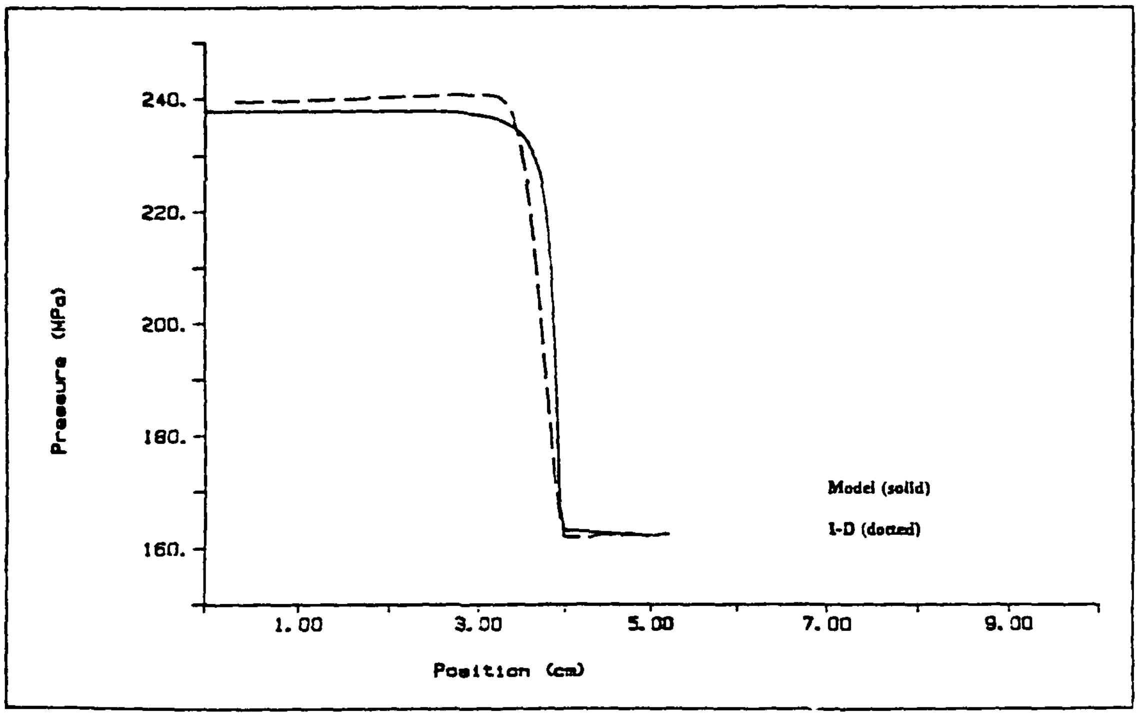

Figure 14. Comparison of Pressure Distributions from the Lagrange Pressure Distribution with Area Change Model (Solid Line) and a 1-Dimensional Simulation (Dotted Line) at Mid-Stroke.

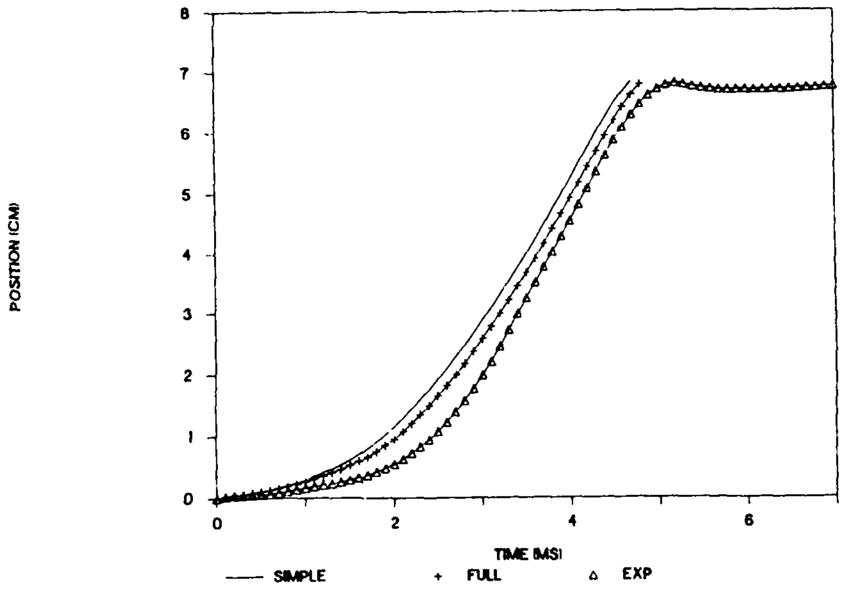

Figure 15. Comparison of the Simple and Full Models with Experimental Piston Position from a 30-mm, Concept VI RLPG (BRL Round 8).

In Figure 15 a comparison of piston position shows that, although the full model is more accurate than the simple model in comparison to experimental piston position, neither reflect the delay from approximately 0.0 to 2.0 ms in establishing the experimental piston velocity.

The comparison of the simple and full simulations of liquid pressure with experimental liquid pressure is shown in Figure 16. Although the first oscillation in liquid pressure is reflected in both models, the oscillations in the two injection models simply become damped too quickly, showing the most significant departure from 1.0 to 2.5 ms where the experimental liquid pressure is noticeable higher than the simulations. It is in this regime that the piston position is also not accurate. The higher liquid pressure and slowed piston may indicate that the flow of liquid propellant has been disturbed in the experiment. In operation the flow of liquid propellant unseats the O-ring into the combustion chamber opening the vent. If the O-ring did not unseat uniformly, it could block the initial flow of liquid leading to higher liquid pressure than predicted and a correspondingly slower piston. The difference in magnitude of the pressure oscillations also indicate that the model may not have an accurate value for the bulk modulus. Once the oscillations have damped, however, both models agree with experimental values through the steady state regime with some deviation near end of stroke. The full model more accurately captures the end of stroke and follows the experimental liquid pressure through the decrease just before the gauge fails.

The calculated values of the discharge coefficient are shown in Figure 17 for the simple and full models in comparison with the values of the discharge coefficient derived, not directly measured, from experimental data. As expected, the values of the simulated discharge coefficients follow the general observations for the piston. Both exhibit initial wide fluctuations and establishment of a

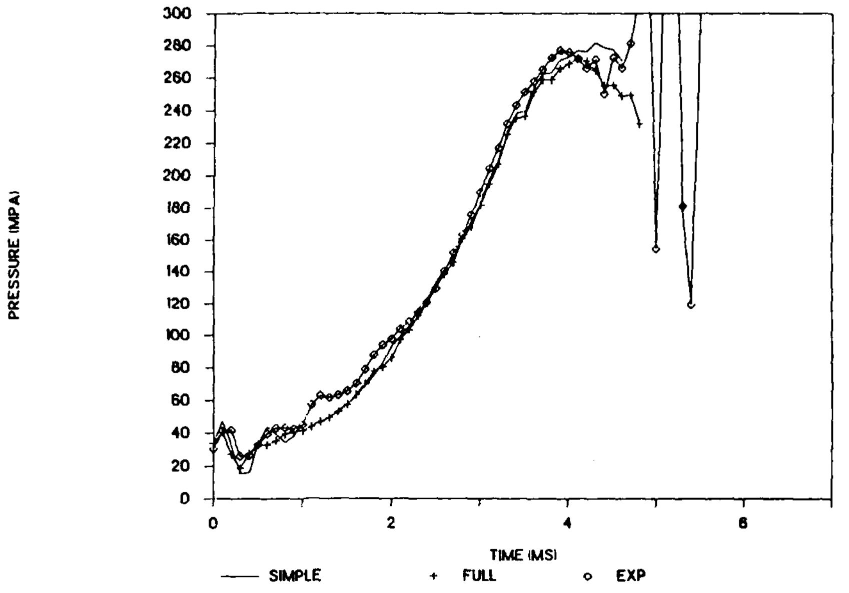

Figure 16. <u>Comparison of the Simple and Full Injection Models with Experimental Liquid Pressure from a 30-mm, Concept VI RLPG (BRL Round 8).</u>

steady state value by 1.0 ms. From 1.0 ms to 2.0 ms the values for the full model are lower than for the simple model reflecting the slower piston observed in Figure 15. The mean values of both models are approximately 0.9 which agree well with the calculated mean values of the discharge coefficient from experiment as shown in Figure 17. However, the inability of the model to predict a time varying discharge coefficient which is predominantly a monotonically increasing function over the first few milliseconds of the ballistic event reflects indirectly the inability of either model to capture the slow piston over the same time period. It may well be that Round 8 represents an anomaly in operation of the liquid propellant gun. The characteristic of a ballistically long, slow rise to steady state is not seen in data from RLPGs with the Belleville springs removed. The use of Belleville springs was a convenient method of breaking the initial seal between the inner piston and bolt in early test fixtures. However, most fixtures built after 1985 have incorporated metal to metal seals and utilized dampers with water or oil to modulate the piston motion.

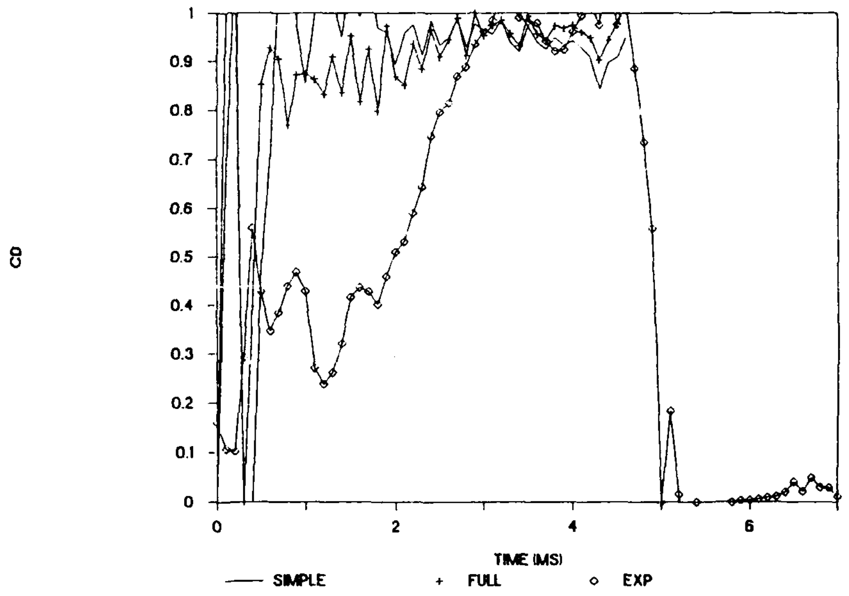

Figure 17. Comparison of the Simple and Full Models with Derived Values of the Discharge Coefficient from a 30-mm, Concept VI RLPG (BRL Round 8).

In general, the full model represents an improvement over the simple model in its ability to more accurately predict the startup regime and end of stroke in a 30-mm, Concept VI RLPG. However, both the simple and full models are in reasonable qualitative and quantitative agreement with experiment, and, in fact, the simple model captures the injection process in the 30-mm RLPG without Belleville springs as exhibited by Shot 7.

VII. CONCLUSIONS

A model of liquid injection in a regenerative liquid propellant gun has been developed which couples the motion of the regenerative piston to the flow of liquid propellant from the reservoir into the combustion chamber. The model is based on a generalization of the Lagrange approximation to address the variation of fluid mass in the reservoir during the ballistic cycle; the variation of area with position in the reservoir; and the variation of area with time at a fixed position in reservoir due to the rearward motion of the contoured injection piston. It is applicable to Concept VI and Concept VIA geometries with a stationary center bolt. Two versions of the model have been considered: a simple model which utilizes a simplified statement of the pressure gradient in the liquid reservoir; and a full model which extends the pressure gradient to consider the contours of the piston head and injection orifice. The following conclusions about the model have been presented in this paper.

1) Compared to experimental data, the simple model adequately describes the motion of the regenerative piston and liquid pressure history for the Concept VIA fixture with a damper and without Belleville springs. The predicted values of the discharge coefficient are in reasonable agreement with experimentally derived mean values.

2) The simple model does not accurately capture the start-up regime of the Concept VI fixture with Belleville springs. In addition, the simple model does not predict the slow rise of the discharge coefficient to steady state observed in experiment.

3) The full model, while displaying better agreement than the simple model to the Concept VI experimental piston position and liquid pressure, does not

predict the slow rise to steady state observed in the experiment. The reasons for the lack of agreement remain speculative. The predicted values of the discharge coefficient are in close agreement with those obtained from the simple model.

4) The predicted mean value of the discharge coefficient is 0.95, a value which agrees with experiment.

In general, the model appears to be an improved description of the liquid injection process for the regenerative liquid propellant gun in comparison with current lumped parameter models, since it does not require empirically determined parameters.

REFERENCES

1. Gough, P.S., "A Model of the Interior Ballistics of Hybrid Liquid Propellant Guns," BRL Contract Report No. BRL-CR-566, March 1987.

2. Pate, R.A. and Schlermen, C.P., "Flow Properties of LGP 1846 at Reduced Temperatures," Proceedings of the 22nd JANNAF Combustion Meeting, October 1985.

3. Coffee, T.P., "The Analysis of Experimental Measurements on Liquid Regenerative Guns," BRL Technical Report No. BRL-TR-2731, May 1986.

4. Morrison, W.F. and Wren, G.P., "A Model of Liquid Injection in a Regenerative Liquid Propellant Gun," BRL Technical Report No. BRL-TR-2851, July 1987.

5. Wren, G.P. and Morrison, W.F., "Velocity and Pressure Distributions in the Liquid Reservoir in a Regenerative Liquid Propellant Gun," BRL Technical Report No. BRL-TR-2933, September 1988.

6. Morrison, W. F. and Wren, G. P., "A Model of Liquid Injection in a Liquid Propellant Gun," Proceedings of the 25th JANNAF Combustion Meeting, October 1988.

7. Wren, G.P. and Morrison, W.F., "Extension of a Model of Liquid Injection in a Regenerative Liquid Propellant Gun Based Upon Comparison witn Experimental Results," 25th JANNAF Combustion

Meeting, October 1988.

8. Schlerman, C., General Electric Corp., private communication.

9. Kanaher, D.K., National Bureau of Standards, private communication.

10. Coffee, T., "Numerical Modeling of Injection in a Liquid Regenerative Propellant Gun," Ballistic Research Laboratory Technical Report, BRL-TR-2897, March 1988.

LIST OF SYMBOLS

$A(x,t)$	Cross Sectional Area of the Flow
$a(x,t)$	$\pi R_0^2 - \pi R^2(x,t)$
$A_L(t)$	$A_R + A_3(t)$
A_D	Cross Sectional Area of Damper Side of Piston
A_p	Cross Sectional Area of Chamber Side of Piston
A_R	Cross Sectional Area of Reservoir Side of Piston
A_T	Cross Sectional Area of Transducer Block
$A_3(t)$	Cross Sectional Area of Injection Orifice
$H(x)$	Heavyside Function
$J_n(x,t)$	Non-Dimensional Integral Functions Arising in Model Equations
$J_n^{03}(t)$	$J_n(s_3,t)$
$\overline{J_n^{03}(t)}$	Space Mean Value of $J_n(x,t)$
$L_n(x,t)$	Function with Units of Length Arising in Model Equations

$L_a^{03}(t)$	$L_a(s_3,t)$
$\overline{L_a^{03}(t)}$	Space Mean Value of $L_a(x,t)$
$L_a^{eff}(t)$	Effective Length Coefficient
$L_v^{eff}(t)$	Effective Length Coefficient
l_1	Initial value of $s_1 - s_0$
$l_1(t)$	$s_1 - s_0$
l_2	$s_3 - s_2$
M_p	Piston Mass
M_T	Transducer Block Mass
$M_p^{eff}(t)$	Effective Piston Mass
$m_L(t)$	Liquid Mass
$m_L^{orifice}$	Liquid Mass in Orifice
$m_L^{eff}(t)$	Effective Liquid Mass
$P(x,t)$	Liquid Pressure
$P_D(t)$	Pressure in the Damper

P_I	Initial Liquid Pressure
$P_0(t)$	Liquid Pressure at Transducer Block
$P_3(t)$	Combustion Chamber Pressure
$\overline{P}(t)$	Space Mean Pressure in Liquid
$R(x,t)$	Radius of Inner Surface of Piston
R_0	$R(0,t)$
$r_b(x)$	Radius of Bolt
$s(x,t)$	Point on Inner Surface of Piston at Position x at Time t
$U^2(t)$	Function with Units of Velocity Squared
$U^2A(t)$	Function with Units of Velocity Squared times Area
u_p	Velocity of Piston
$V_R(t)$	Volume of Reservoir
$v(x,t)$	Liquid Velocity
v_3	Liquid Velocity at Orifice Exit
$\overline{v(t)}$	Space Mean Liquid Velocity

$\overline{v^2(t)}$	Space Mean Average of Square of Liquid Velocity
$\overline{vl(t)}$	Space Mean Average of $\int_0^x v(x',t)dx'$
ρ	Liquid Density
ψ	Liquid Loss through Orifice

APPENDIX A

DEFINITION OF TERMS

INTENTIONALLY LEFT BLANK.

The ordinary differential equations given by Equations (6) and (7) in the full model involve a number of geometric integrals. These integrals have been analytically simplified, when possible, for the specific geometry of the Concept VI. However, it is necessary to evaluate some expressions numerically at each timestep since area is a function of both piston position and time. The complete derivation has been presented in earlier reports, and a summary is provided here of those expressions necessary to solve the system of ordinary differential equations.

$$L_u^{eff}(t) = \frac{A_R}{A_L} L_2^{03}(t) - L_1^{03}(t) \tag{A1}$$

$$L_v^{eff}(t) \doteq \frac{A_3}{A_L} L_2^{03}(t) \tag{A2}$$

$$\begin{aligned} U^2(t) \doteq\; & u_p^2\left[\left(\overline{J_1^{03}(t)} - J_1^{03}(t)\right) + \left(\overline{J_3^{03}(t)} - J_3^{03}(t)\right) + \frac{1}{2} J_6^{03}(t)\right] \\ & + u_p\left(u_p \frac{A_R}{A_L} - v_3 \frac{A_3}{A_L}\right)\left[\left(J_2^{03}(t) - \overline{J_2^{03}(t)}\right) - \left(J_5^{03}(t) - \overline{J_5^{03}(t)}\right) - J_7^{03}(t)\right] \\ & - u_p v_3\left[J_4^{03}(t) - \overline{J_4^{03}(t)}\right] + \frac{1}{2}\left(u_p \frac{A_R}{A_L} - v_3 \frac{A_3}{A_L}\right)^2 J_8^{03}(t) \\ & + \dot{u}_p\left[\overline{L_1^{03}(t)} - \frac{A_R}{A_L}\overline{L_2^{03}(t)}\right] + \dot{v}_3\left[\frac{A_3}{A_L}\overline{L_2^{03}(t)}\right] \end{aligned} \tag{A3}$$

$$M_p^{eff}(t) = M_p\left\{1 + \frac{m_L}{M_P}\left[\frac{A_R}{V_R} L_3^{03}(t) - J_9^{03}(t)\right]\right\} \tag{A4}$$

$$m_L^{eff}(t) = m_L\left\{\frac{A_3}{V_R} L_3^{03}(t)\right\} \tag{A5}$$

$$
\begin{aligned}
U^2A(t) = \; & A_T\{u_p^2\left[\overline{J_1^{03}(t)} + \overline{J_3^{03}(t)} + \frac{1}{2}J_6^{03}(t)\right] \\
& - u_p\left(u_p\frac{A_R}{A_L} - v_3\frac{A_3}{A_L}\right)\left[\overline{J_2^{03}(t)} - \overline{J_5^{03}(t)} + J_7^{03}(t)\right] \\
& + u_pv_3[\overline{J_4^{03}(t)}] + \frac{1}{2}\left(u_p\frac{A_R}{A_L} - v_3\frac{A_3}{A_L}\right)^2[J_8^{03}(t)] \\
& + u_p\left[\overline{L_1^{03}(t)} - \frac{A_R}{A_L}\overline{L_2^{02}(t)}\right] + v_3\left[\frac{A_3}{A_L}\overline{L_2^{03}(t)}\right]\} \\
& - u_p(u_pA_R + v_3A_3) - u_pv_3\left[L_3^{03}(t)\frac{\partial\pi r_b^2(x)}{\partial x}\bigg|_{s_3}\right] \\
& - 2u_p(u_pA_R - v_3A_3)J_9^{03}(t) \\
& + \left(\frac{u_pA_R - v_3A_3}{V_R}\right)[(u_pA_R - v_3A_3) + u_pA_L]L_3^{03}(t) \; . \qquad (A6)
\end{aligned}
$$

The terms involving J and the mean quanities denoted by the bar are defined below.

$$J_1(x,t) = \int_0^x \frac{a(x',t)}{A^2(x',t)} \frac{\partial a(x',t)}{\partial x'}dx' \qquad (A7)$$

$$J_2(x,t) = \int_0^x \frac{V(x',t)}{V_R} \frac{A_L}{A^2(x',t)} \frac{\partial a(x',t)}{\partial x'}dx' \qquad (A8)$$

$$J_3(x,t) = \int_0^x \frac{1}{A(x',t)} \frac{\partial a(x',t)}{\partial x'}dx' \qquad (A9)$$

$$J_4(x,t) = \frac{\partial\pi r_b^2(x)}{\partial x}\bigg|_{s_3} \int_0^x \frac{V(x',t)}{V_R} \frac{dx'}{A(x',t)} \qquad (A10)$$

$$J_5(x,t) = \frac{A_L}{V_R}\int_0^x \left[\frac{a(x',t)}{A(x',t)} - \frac{A_L}{A(x',t)} \frac{V(x',t)}{V_R}\right]dx' \qquad (A11)$$

$$J_6^{03}(t) = \frac{1}{V_R} \int_0^{s_3} \frac{a^2(x,t)}{A(x,t)} dx \tag{A12}$$

$$J_7^{03}(t) = \frac{A_L}{V_R} \int_0^{s_3} \frac{V(x,t)}{V_R} \frac{a(x,t)}{A(x,t)} dx \tag{A13}$$

$$J_8^{03}(t) = \frac{A_L}{V_R} \int_0^{s_3} \left[\frac{V(x,t)}{V_R}\right]^2 \frac{A_L}{A(x,t)} dx \ . \tag{A14}$$

$$J_9^{03}(t) = \frac{1}{V_R} \int_0^{s_3} a(x,t) dx \tag{A15}$$

$$\overline{J_1^{03}(t)} = \frac{1}{V_R} \int_0^{s_3} A(x,t) J_1(x,t) dx \tag{A16}$$

$$\overline{J_2^{03}(t)} = \frac{1}{V_R} \int_0^{s_3} A(x,t) J_2(x,t) dx \tag{A17}$$

$$\overline{J_3^{03}(t)} = \frac{1}{V_R} \int_0^{s_3} A(x,t) J_3(x,t) dx \tag{A18}$$

$$\overline{J_4^{03}(t)} = \frac{1}{V_R} \int_0^{s_3} A(x,t) J_4(x,t) dx \tag{A19}$$

$$\overline{J_5^{03}(t)} = \frac{1}{V_R} \int_0^{s_3} A(x,t) J_5(x,t) dx \tag{A20}$$

$$L_1(x,t) = \int_0^{x} \frac{a(x',t)}{A(x',t)} dx' \tag{A21}$$

$$L_2(x,t) = \frac{A_L}{V_R} \int_0^{x} \frac{V(x',t)}{A(x',t)} dx' \ . \tag{A22}$$

$$\overline{L_1^{03}(t)} = \frac{1}{V_R} \int_0^{s_3} A(x,t) L_1(x,t) dx \tag{A23}$$

$$\overline{L_2^{03}(t)} = \frac{1}{V_R} \int_0^{s_3} A(x,t) L_2(x,t) dx \tag{A24}$$

$$L_3^{03}(t) = \frac{1}{V_R} \int_0^{s_3} V(x,t) dx \quad . \tag{A25}$$

Note that the integrals J_1 through J_9 together with their mean values are dimensionless, while the integrals L_1 and L_2 and their mean values have units of length.

APPENDIX B

MODEL INPUT FOR 30-MM, CONCEPT VIA

INTENTIONALLY LEFT BLANK.

The input file listed below is descriptive of the 30-mm, Concept VIA geometry used in GE Shot 7. The test fixture was built and fired by the General Electric Company, Tactical Systems Department, under contract DAAK11-84-C-0055.

```
RLPLCH--30MM--CONCEPT 6A SLICE--MODEL OF GE DATA
45.5125 35.4957 23.7640   AC  AP  AR   (CM**2)
1.397       1.6794        RLPRIME  RLVENT  (CM)
0.0                       OFFSET (CM) OF PISTON FROM END OF BLT
2012.0                    PMASS   (G)
83.25821                  VOLFO   (CM**3)
2                         TVENT--ACTUAL PISTON
2.159                     RLTEMP--(CM)--ORIGINAL
1.437  5661.1  9.2649     RHOL      RK1  RK2
0                         IFRL--FRICTION LOSS OPTION OF LIQ
0                         IFRP--FRICTION LOSS OPTION OF PIS
D:CHAMB6A.DAT
D:GEOMC6A.GE
D:GE6A.GRA
1.46  0. 0. 0.0           PRES      VELH  UPISTON  XPISTON
0.0001 .00001             TINC      EPS
1 0 0                     METH      MITER  KWRITE
1                         DIFEQN--DIFF EQN SET
1.78562 2.0665782 3.27914 RP3  RP2  RP1  (CM)--PISTON
1.78562 1.55575 1.55575   RB3  RB2  RB1  (CM)--PISTON
21.2068                   VOL12  (CM**3)
5.5231896                 VOL23  (CM**3)
0                         IWRITE
0                         PRESSURE DISTRIBUTION OPTION
0.5588      0.76          RLBLTF (BOLT FLAT),RLBLTS (BOLT SLT)
20  1                     NPRPTS,IP (NO. PTS,OPEN GRAPH FILE)
D:DISTVP2.GRA
```

INTENTIONALLY LEFT BLANK.

APPENDIX C

MODEL INPUT FOR 30-MM, CONCEPT VI

INTENTIONALLY LEFT BLANK.

The input file listed below is descriptive of the 30-mm, Concept VI geometry used in BRL Round 8. The test fixture was built by the General Electric Company, Tactical Systems Department, under contract DAAK11-84-C-0055 and fired at the Ballistic Research Laboratory.

```
RLPLCH--ROUND 8--30MM
44.847 34.326 23.278          AC  AP  AR   (CM**2)
1.432 1.04                    RLPRIME  RLVENT  (CM)
0.544                         OFFSET (CM) OF PISTON FROM END OF BLT
2109.2                        PMASS   (G)
172.62764                     VOLFO   (CM**3)
2                             TVENT--ACTUAL PISTON
5.94680                       RLTEMP--(CM)--ORIGINAL
1.437  5350.0  9.11           RHOL RK1  RK2
0                             IFRL--FRICTION LOSS OPTION OF LIQ
0                             IFRP--FRICTION LOSS OPTION OF PIS
ptoff64.dat
r8geo55.dat
jannaf88.gra
29. 0. 358. 0.00              PRES VELH  UPISTON  XPISTON
0.0001 .00001                 TINC EPS
1 0 0                         METH MITER  KWRITE
1                             DIFEQN--DIFF EQN SET
1.83 1.83 3.28                RP3  RP2  RP1  (CM)--PISTON
1.7977 1.7977 1.65            RB3  RB2  RB1  (CM)--PISTON
17.908367                     VOL12  (CM**3)
2.0266094                     VOL23  (CM**3)
0.5588 0.76                   RLBLTF (BOLT FLAT),RLBLTS (BOLT SLT)
0                             IWRITE
0                             PRESSURE DISTRIBUTION OPTION
0.5588 0.76                   RLBLTF (BOLT FLAT),RLBLTS (BOLT SLT)
20  1                         NPRPTS,IP (NO. PTS,OPEN GRAPH FILE)
distvpll.gra
```

INTENTIONALLY LEFT BLANK.

APPEDIX D

LISTING OF THE FULL VERSION OF THE CODE

INTENTIONALLY LEFT BLANK.

```
      PROGRAM INJECT
C     ***** RLP LIQUID CHAMBER *****
C
C     PROGRAM TO EXAMINE THE DISCHARGE COEFFICIENT DESCRIBING
C     LIQUID PROPELLANT FLOW IN A REGENERATIVE LIQUID PROPELLANT
C     GUN BY COUPLING MOTION OF REGENERATIVE PISTON AND LIQUID
C
      DIMENSION Y(4),YDOT(4),WK(1000)
      COMMON /NAMES/ TITLE,DATA,GRAF,GEOM,VPDIST
      COMMON /FI2/ AC,AP,AR,AL,IDIFEQN
      COMMON /FI3/ RLPRIME,RLVENT,RLMAX,RLTEMP,RLT
      COMMON /FI4/ PMASS,OFFSET
      COMMON /FI5/ TVENT,NROD,SROD(20),RROD(20),AROD(20),JFLAG
      COMMON /FI6/ RHOL,RK1,RK2,VOLFO,PSI,AV
      COMMON /FI7/ IFRP,NOPT,NFLOSS,FRICPOS(20),FLOSS(20),NFIT
      COMMON /FI8/ TI(500),CCP(500),FLAG
      COMMON /FI9/ PRESRES,VELH,UPISTON,XPISTON
      COMMON /F10/ TINC,EPS,TOUT
      COMMON /F11/ METH,MITER,KWRITE
      COMMON /F12/ TMS,RMASSL,PRESRESN,PRESCH,RHOLN,BRPRES
      COMMON /F13/ NPTS,IWRITE
      COMMON /F14/ CD,CDC,RKNVIS
      COMMON /F15/ NBFLAG,NPFLAG,NPIST,SPIST(20),RPIST(20),APIST(20)
      COMMON /F16/ XPIST(1000),VOLF(1000),AVT(1000),ALIQ(1000),NGPTS
      COMMON /F17/ XPISTC,IXPISTC,RLPRIMEO
      COMMON /F18/ ISET,RM,RINTVS12
      COMMON /F19/ RP1,RP2,RP3,RB1,RB2,RB3,VOL12,VOL23
      COMMON /F20/ IFRL,RE,NGAP,SGAP(50),GAPW(50)
      COMMON /F21/ RLBLTF,RLBLTS,NPRPTS,ORIGS3,ORIGS2,ORIGS1,X3,X2,X1,IP
      COMMON /F22/ CUPISTON,CVELH,CXPISTON,CRMASSL
      COMMON /F23/ IPRES,RADDP,RADDB,RATIOP,RATIOB
      COMMON /F24/ S3,S2,S1,RINTAR,RINTP3
C
      CHARACTER*80 TITLE
      CHARACTER*80 DATA
      CHARACTER*80 GEOM
      CHARACTER*80 GRAF
      CHARACTER*80 VPDIST
      INTEGER TVENT,FLAG
C
      EXTERNAL F
C
C     INITIALIZING COUNTERS FOR ARRAYS AND ENTRY POINTS
      NPTS=0
      FLAG=1
      NBFLAG=1
      NPFLAG=1
      JFLAG=1
      IXPISTC=0
      ISET=0
C
C     OUTPUT TO FILE GIVEN ON COMMAND LINE
C
C     READ INPUT FORM LIST-DIRECTED BATCH RUN
C              AND ECHO TO PRINTER
      CALL INPUT
```

```
C
C     READ TIME-C CH PRES BOUNDARY CONDITIONS
      CALL PRESCCH(1)
C
C     READ GEOMETRY
      IF (TVENT.EQ.2) CALL BOLT2(1)
C
C     SET INITIAL CONDITIONS
      CALL STARTUP(Y,YDOT,N)
C
C     SET INTEGRATOR PARAMETERS
      TO=0.0
      TOUT=TINC
      TMS=0.0
      HO=1.0E-03
      INDEX=1
      MSTATE=1
      LENW=1000
C
C     FIRST CALL TO DIFFERENTIAL EQUATIONS
      CALL F(N,0.0,Y,YDOT)
C     FIRST LINE OF OUTPUT
      CALL CAPTION(IDIFEQN)
      CALL OUT (Y,YDOT)
  200 CALL SDRIV1 (N,TO,Y,TOUT,MSTATE,EPS,WK,LENW)
C
C     DIAGNOSTICS BASED ON IER
      IF (MSTATE.GT.2) THEN
         WRITE (*,500) MSTATE
  500 FORMAT (//,' FATAL ERROR--MSTATE=',I5)
         WRITE (*,505) EPS,N,Y(1),Y(2),Y(3),Y(4)
  505 FORMAT(' EPS,N,Y(1),Y(2),Y(3),Y(4):',F10.5,I3,4F15.5)
         WRITE (*,515) TOUT,TO
  515 FORMAT(' TOUT,TO:',2F10.5)
         WRITE (*,525) YDOT(1),YDOT(2),YDOT(3),YDOT(4)
  525 FORMAT (' YDOT(1),YDOT(2),YDOT(3),YDOT(4):',4F15.5)
         STOP
      ENDIF
C
C     DIAGNOSTICS BASED ON KWRITE
      IF (KWRITE.EQ.1) THEN
         WRITE (*,505) EPS,N,Y(1),Y(2),Y(3),Y(4)
         WRITE (*,515) TOUT,TO
         WRITE (*,525) YDOT(1),YDOT(2),YDOT(3),YDOT(4)
      ENDIF
C
C     NOT ALL LIQUID CAN BE INJECTED
C     SINCE BEHIND AND UNDER PISTON
C     APPROXIMATE LIQUID LEFT AS 2.76 CC
C     IF LIQUID MASS=2.76 OR PISTON TRAVEL=MAX TRAVEL
C     STOP THE INTEGRATOR
C
C     POSSIBLE PISTON TRAVEL REMAINING (PTR)
      PTR=RLMAX-XPISTON
C
      IF ((VOLFO.LE.2.76).OR.(PTR.LE.0.01)) GOTO 300
```

```
C     TIME (TOUT) IN SECONDS FOR INTEGRATOR
C     PRINTOUT (TMS) IN MILLISECONDS
      TMS-TOUT*1.0E3
      CALL OUT (Y,YDOT)
      TOUT-TOUT+TINC
      GOTO 200
C
C 300 IF ((VOLFO.GT.2.76).AND.(PTR.LE.0.01)) WRITE(*,400)
C     IF ((PTR.GT.0.01).AND.(VOLFO.LE.2.76)) WRITE(*,410)
C
  300 TMS-TMS+TINC*1.0E3
      CALL OUT (Y,YDOT)
      WRITE (*,320) NPTS
  320 FORMAT (//,' NUMBER OF POINTS IN GRAPHIC FILE-',I5)
C
C     ERROR MESSAGES
C 400 FORMAT (//,' LIQUID MASS REMAINS BUT PISTON TRAVEL COMPLETED')
C 410 FORMAT (//,' PISTON TRAVEL NOT COMPLETED BUT NO LIQUID VOLUME')
C
C     CLOSING FILES
      CLOSE (16)
      CLOSE (17)
      CLOSE (19)
      STOP
      END
C     *****************************************************
      SUBROUTINE INPUT
C     DESCRIPTION OF LIQUID CHAMBER GEOMETRY
      COMMON /NAMES/ TITLE,DATA,GRAF,GEOM,VPDIST
      COMMON /FI2/ AC,AP,AR,AL,IDIFEQN
      COMMON /FI3/ RLPRIME,RLVENT,RLMAX,RLTEMP,RLT
      COMMON /FI4/ PMASS,OFFSET
      COMMON /FI5/ TVENT,NROD,SROD(20),RROD(20),AROD(20),JFLAG
      COMMON /FI6/ RHOL,RK1,RK2,VOLFO,PSI,AV
      COMMON /FI7/ IFRP,NOPT,NFLOSS,FRICPOS(20),FLOSS(20),NFIT
      COMMON /FI8/ TI(500),CCP(500),FLAG
      COMMON /FI9/ PRESRES,VELH,UPISTON,XPISTON
      COMMON /F10/ TINC,EPS,TOUT
      COMMON /F11/ METH,MITER,KWRITE
      COMMON /F12/ TMS,RMASSL,PRESRESN,PRESCH,RHOLN,BRPRES
      COMMON /F13/ NPTS,IWRITE
      COMMON /F14/ CD,CDC,RKNVIS
      COMMON /F15/ NBFLAG,NPFLAG,NPIST,SPIST(20),RPIST(20),APIST(20)
      COMMON /F16/ XPIST(1000),VOLF(1000),AVT(1000),ALIQ(1000),NGPTS
      COMMON /F17/ XPISTC,IXPISTC,RLPRIMEO
      COMMON /F18/ ISET,RM,RINTVS12
      COMMON /F19/ RP1,RP2,RP3,RB1,RB2,RB3,VOL12,VOL23
      COMMON /F20/ IFRL,RE,NGAP,SGAP(50),GAPW(50)
      COMMON /F21/ RLBLTF,RLBLTS,NPRPTS,ORIGS3,ORIGS2,ORIGS1,X3,X2,X1,IP
      COMMON /F22/ CUPISTON,CVELH,CXPISTON,CRMASSL
      COMMON /F23/ IPRES,RADDP,RADDB,RATIOP,RATIOB
C
      CHARACTER*80 TITLE
      CHARACTER*80 DATA
      CHARACTER*80 GEOM
      CHARACTER*80 GRAF
      CHARACTER*80 VPDIST
      INTEGER TVENT,FLAG
C
C     INPUT FROM FILE GIVEN ON COMMAND LINE
      READ (*,150) TITLE
      WRITE (*,150) TITLE
C     DESCRIPTION OF LIQUID CHAMBER
      READ (*,*) AC,AP,AR
      WRITE (*,20) AC
      WRITE (*,30) AP
      WRITE (*,40) AR
      READ (*,*) RLPRIME,RLVENT
      WRITE (*,50) RLPRIME
      RLPRIMEO=RLPRIME
      WRITE (*,60) RLVENT
      READ (*,*) OFFSET
      WRITE (*,65) OFFSET
      READ (*,*) PMASS
      WRITE (*,70) PMASS
      READ (*,*) VOLFO
      WRITE (*,120) VOLFO
C     DESCRIPTION OF BOLT
C     IF TVENT=1 THEN STRAIGHT BOLT
C     IF TVENT=2 THEN REAL BOLT DESCRIBED BY RADIUS
C     TVENT=2 USES SEPARATE FILE
      READ (*,*) TVENT
      WRITE (*,80) TVENT
      IF (TVENT.EQ.1) CALL BOLT1(1)
      IF (TVENT.EQ.2) THEN
         READ (*,*) RLTEMP
         WRITE (*,82) RLTEMP
         RLMAX=RLTEMP+RLPRIME
         WRITE (*,84) RLMAX
      ENDIF
C     LIQUID PROPELLANT PROPERTIES
      READ (*,*) RHOL,RK1,RK2
      WRITE (*,90) RHOL
      WRITE (*,100) RK1
      WRITE (*,110) RK2
      RK1=RK1*1.0E7
C     FRICTION LOSS OF LIQUID (IFRL) AND PISTON (IFRP)
      READ (*,*) IFRL
      WRITE (*,130) IFRL
      IF (IFRL.EQ.1) CALL FRIC1(0,RLIQLOS)
      READ (*,*) IFRP
      WRITE (*,140) IFRP
      IF (IFRP.EQ.1) CALL FRIC2(0,RPISLOS)
C     NAME OF FILE IN WHICH TIME-PR BOUNDARY COND STORED
      READ (*,160) DATA
      WRITE (*,170) DATA
C     NAME OF FILE IN WHICH GEOMETRY DATA STORED
      IF (TVENT.EQ.2) THEN
         READ (*,160) GEOM
         WRITE (*,200) GEOM
      ENDIF
C     NAME OF FILE IN WHICH GRAPH DATA STORED
```

```
      READ (*,160) GRAF
      WRITE (*,180) GRAF
C     INITIALIZE
      CALL INITIAL
C     PARAMETERS FOR INTEGRATOR
      CALL NUMERIC
C     DIFFERENTIAL EQUATION SET
C     SET 1:  ORIGINAL 8/1/86
C     SET 2:  REVISED 8/20/86
C     SET 3:  BASELINE WITHOUT INERTIAL TERMS
      READ (*,*) IDIFEQN
      WRITE (*,190) IDIFEQN
C     IF SET 1, INPUT RADIUS OF PISTON AT STATIONS 1,2,3
C                 INPUT RADIUS OF BOLT AT STATIONS 1,2,3
C                 INPUT VOLUME ENCLOSED FROM 1-2 AND 2-3
C     IF IWRITE=1, DIAGNOSTICS OUTPUT
      IF ((IDIFEQN.EQ.1).OR.(IDIFEQN.EQ.2)) THEN
         READ(*,*) RP3,RP2,RP1
         WRITE(*,230) RP3,RP2,RP1
         READ(*,*) RB3,RB2,RB1
         WRITE(*,240) RB3,RB2,RB1
         READ(*,*) VOL12
         WRITE(*,250) VOL12
         READ(*,*) VOL23
         WRITE(*,260) VOL23
         READ(*,*) RLBLTF,RLBLTS
         WRITE(*,280) RLBLTF,RLBLTS
         READ(*,*) IWRITE
         WRITE(*,270) IWRITE
      ENDIF
C     IF SET 3, INPUT REYNOLDS NUMBER AND TABLE OF GAP FOR VENT
      IF (IDIFEQN.EQ.3) THEN
         READ (*,*) RKNVIS
         WRITE (*,210) RKNVIS
         READ (*,*) NGAP
      DO 18 I=1,NGAP
         READ (*,*) SGAP(I),GAPW(I)
         WRITE (*,220) SGAP(I),GAPW(I)
   18 CONTINUE
      ENDIF
C     OPTION FOR PRESSURE DISTRIBUTION
      READ (*,*) IPRES
      WRITE (*,145) IPRES
      IF (IPRES.EQ.1) CALL PRESDIS(1)

   20 FORMAT (//,' COMBUSTION CHAMBER AREA =',F12.5)
   30 FORMAT (' PISTON AREA--C CH SIDE  =',F12.5)
   40 FORMAT (' PISTON AREA--RES SIDE   =',F12.5)
   50 FORMAT (' LENGTH L PRIME          =',F12.5)
   60 FORMAT (' LENGTH OF VENT          =',F12.5)
   65 FORMAT (' PISTON OFFSET           =',F12.5)
   70 FORMAT (' PISTON MASS             =',F12.5)
   80 FORMAT (//,' VENT OPTION             =',I6)
   82 FORMAT (' STRAIGHT LENGTH OF PIST =',F12.5)
   84 FORMAT (' MAX PISTON TRAVEL       =',F12.5)
   90 FORMAT (//,' DENSITY LIQUID          =',F12.5)
```

```
  100 FORMAT (' K1                      =',F12.5)
  110 FORMAT (' K2                      =',F12.5)
  120 FORMAT (' VOLUME LIQUID           =',F12.5)
  130 FORMAT (//,' FRICTION LOSS LIQ OPTION=',I6)
  140 FORMAT (//,' FRICTION LOSS PIS OPTION=',I6)
  145 FORMAT (//,' PR DISTRIBUTION OPTION  =',I6)
  150 FORMAT(A)
  160 FORMAT(A)
  170 FORMAT (//,' TIME-C CH PRES DATA FILE: ',A)
  180 FORMAT (//,' GRAPH DATA FILE:          ',A,//)
  190 FORMAT (//,' DIFFERENTIAL EQUATION SET.',I2)
  200 FORMAT (//,' GEOMETRY DATA FILE:       ',A)
  210 FORMAT (//,' KINEMATIC VISCOSITY      =',F12.5,/)
  220 FORMAT (' POSITION=',F12.5,10X,' GAP=',F12.5)
  230 FORMAT (//,' RAD PIST3 =',F10.5,5X,'RAD PIST2 =',F10.5,5X,
     + 'RAD PIST1 =',F10.5)
  240 FORMAT (//,' RAD BOLT3 =',F10.5,5X,'RAD BOLT2 =',F10.5,5X,
     + 'RAD BOLT1 =',F10.5)
  250 FORMAT (//,' VOL FUEL12=',F10.5)
  260 FORMAT (' VOL FUEL23=',F10.5,/)
  270 FORMAT (' IWRITE     =',I4,//)
  280 FORMAT (' FLAT LEN BOLT=',F12.5,' SLANT LEN BOLT=',F12.5)
      RETURN
      END
C     *****************************************************
      SUBROUTINE BOLT1(IOPT)
C     STRAIGHT BOLT
C     MAX PISTON TRAVEL CHOSEN TO AGREE WITH INITIAL LIQUID VOL
C
      COMMON /FI2/ AC,AP,AR,AL,IDIFEQN
      COMMON /FI3/ RLPRIME,RLVENT,RLMAX,RLTEMP,RLT
      COMMON /FI5/ TVENT,NROD,SROD(20),RROD(20),AROD(20),JFLAG
      COMMON /FI6/ RHOL,RK1,RK2,VOLFO,PSI,AV
      COMMON /FI9/ PRESRES,VELH,UPISTON,XPISTON
      COMMON /F12/ TMS,RMASSL,PRESRESN,PRESCH,RHOLN,BRPRES
C
      INTEGER TVENT
C
C     IF IOPT=2 RECOMPUTE VOLUME TO FIND LIQUID DENSITY
      IF (IOPT.EQ.2) GOTO 100
C
C     OTHERWISE READ FROM BATCH RUNSTREAM AND FIND MAX PISTON TRAVEL
C     AND ORIGINAL LENGTH L (RLTEMP) VARYING WITH TIME
      READ (*,*) AV
      WRITE (*,15) AV
      RLMAX=VOLFO/(AR+AV)
      WRITE (*,10) RLMAX
      RLTEMP=RLMAX-RLPRIME
      WRITE (*,20) RLTEMP
C
C     AREA OF LIQUID
      AL=AR+AV
C
   10 FORMAT ('MAX PISTON TRAVEL       =',F12.5)
   15 FORMAT ('CONSTANT VENT AREA      =',F12.5)
   20 FORMAT ('ORIGINAL L IN RES       =',F12.5)
```

```
      GOTO 200
C
C     RECOMPUTING VOLUME:(MAX TRAVEL-TRAVEL)*(AREA OF LIQUID)
C     SHOULD CONSIDER SLANT ON PISTON HERE--LATER
C
  100 VOLFO=(RLMAX-XPISTON)*(AR+AV)
      RLT=RLTEMP-XPISTON
C
C     LENGTH L(T) COLLAPSED WHEN L.E. 0.0001
      IF (RLT.LE.0.0001) RLT=0.0
C
C     IF RLT COLLAPSED, RLPRIME DECREASES
      IF (RLT.EQ.0.0) RLPRIME=RLMAX-XPISTON
      IF (RLPRIME.LE.0.1) RLPRIME=0.0
C
  200 RETURN
      END
C     ******************************************************
      SUBROUTINE BOLT2(IOPT)
C     BOLT RADIUS AS FUNCTION OF PISTON TRAVEL
C
      COMMON /NAMES/ TITLE,DATA,GRAF,GEOM,VPDIST
      COMMON /FI2/ AC,AP,AR,AL,IDIFEQN
      COMMON /FI3/ RLPRIME,RLVENT,RLMAX,RLTEMP,RLT
      COMMON /FI4/ PMASS,OFFSET
      COMMON /FI5/ TVENT,NROD,SROD(20),RROD(20),AROD(20),JFLAG
      COMMON /FI6/ RHOL,RK1,RK2,VOLFO,PSI,AV
      COMMON /FI7/ IFRP,NOPT,NFLOSS,FRICPOS(20),FLOSS(20),NFIT
      COMMON /FI8/ TI(500),CCP(500),FLAG
      COMMON /FI9/ PRESRES,VELH,UPISTON,XPISTON
      COMMON /F10/ TINC,EPS,TOUT
      COMMON /F11/ METH,MITER,KWRITE
      COMMON /F12/ TMS,RMASSL,PRESRESN,PRESCH,RHOLN,BRPRES
      COMMON /F13/ NPTS,IWRITE
      COMMON /F14/ CD,CDC,RKNVIS
      COMMON /F15/ NBFLAG,NPFLAG,NPIST,SPIST(20),RPIST(20),APIST(20)
      COMMON /F16/ XPIST(1000),VOLF(1000),AVT(1000),ALIQ(1000),NGPTS
      COMMON /F17/ XPISTC,IXPISTC,RLPRIMEO
      COMMON /F18/ ISET,RM,RINTVS12
      COMMON /F19/ RP1,RP2,RP3,RB1,RB2,RB3,VOL12,VOL23
      COMMON /F20/ IFRL,RE,NGAP,SGAP(50),GAPW(50)
C
      CHARACTER*80 TITLE
      CHARACTER*80 DATA
      CHARACTER*80 GEOM
      CHARACTER*80 GRAF
      CHARACTER*80 GEOMID
      CHARACTER*80 VPDIST
      INTEGER TVENT,FLAG
C
C     IF IOPT=2 INTERPOLATE TO FIND VENT AREA, VOLUME FUEL, AREA LIQ
      IF (IOPT.EQ.2) GOTO 150
C
C     OPEN GEOMETRY FILE AND READ ARRAY
C
      OPEN (18,FILE=GEOM,IOSTAT=IOS,ERR=2,STATUS='OLD')
```

```
    2 IF (IOS.NE.0) WRITE (6,3) IOS
    3 FORMAT(' ERROR OPENING GEOM FILE:',I10)
      REWIND(18)
      READ (18,10) GEOMID
      READ (18,*) NGPTS
      DO 50 I=1,NGPTS
         READ (18,*) XPIST(I),VOLF(I),AVT(I),ALIQ(I)
   50 CONTINUE
      CLOSE (18)
      GOTO 200
C
C     IF IOPT=2 READ VOLFO, AV, AL FROM ARRAY
C
C     LOOKUP RADIUS OF PISTON AS MEASURED FROM END OF PISTON
  150 IF (XPISTON.LE.0.0) XPISTON=0.0
      DO 60 I=1,NGPTS
         IF (XPISTON.LE.XPIST(I)) GOTO 65
   60 CONTINUE
      IF (I.GT.NGPTS) THEN
          WRITE (*,62) XPISTON
          STOP
      ENDIF
   62 FORMAT (' ERROR MESSAGE FROM BOLT2--I > NGPTS--XPISTON=',F12.5)
   65 IF (XPISTON.EQ.XPIST(I)) THEN
         VOLFO=VOLF(I)
         AV=AVT(I)
         AL=ALIQ(I)
      ELSE
         VOLFO=VOLF(I-1)+(VOLF(I)-VOLF(I-1))*((XPISTON-XPIST(I-1))/
     +                (XPIST(I)-XPIST(I-1)))
         AV=AVT(I-1)+(AVT(I)-AVT(I-1))*((XPISTON-XPIST(I-1))/
     +             (XPIST(I)-XPIST(I-1)))
         AL=ALIQ(I-1)+(ALIQ(I)-ALIQ(I-1))*((XPISTON-XPIST(I-1))/
     +             (XPIST(I)-XPIST(I-1)))
      ENDIF
C
C     LENGTHS RLT AND RLPRIME
      RLT=RLTEMP-XPISTON
C
C     LENGTH L(T) COLLAPSED WHEN L.E. 0.0001
      IF (RLT.GE.0.0001) THEN
         GOTO 200
      ELSE
         RLT=0.0
         IXPISTC=IXPISTC+1
      ENDIF
C
C     SAVE XPISTC FOR ADJUSTING RLPRIME
      IF (IXPISTC.EQ.1) XPISTC=XPISTON
C
C     IF RLT COLLAPSED, RLPRIME DECREASES
C     RLPRIME=(ORIGINAL RLPRIME)-(DISTANCE MOVED THROUGH RLPRIME)
      RLPRIME=RLPRIMEO-(XPISTON-XPISTC)
C
C     COLLAPSE RLPRIME
      IF (RLPRIME.LE.0.001) RLPRIME=0.0
      IF (RLPRIME.EQ.0.0) THEN
          WRITE (*,180)
          STOP
      ENDIF
  180 FORMAT (' MESSAGE FROM BOLT2--RLPRIME=0--TRAVEL COMPLETE')
C
   10 FORMAT (A)
  200 RETURN
      END
C     ***************************************************
      SUBROUTINE FRIC1(IOPT,RLIQLOS)
C
C     FRICTION LOSS ASSOCIATED WITH LIQUID
C     COMPUTES INLET LOSS
      COMMON /FI3/ RLPRIME,RLVENT,RLMAX,RLTEMP,RLT
      COMMON /FI6/ RHOL,RK1,RK2,VOLFO,PSI,AV
      COMMON /FI9/ PRESRES,VELH,UPISTON,XPISTON
      COMMON /F14/ CD,CDC,RKNVIS
      COMMON /F20/ IFRL,RE,NGAP,SGAP(50),GAPW(50)
C
C     IOPT=0 ON FIRST CALL--READS INPUT
C     IOPT=1 TO COMPUTE LOSS IN DIF EQUATIONS
C
      IF (IOPT.EQ.1) GOTO 50
C
      READ (*,*) PSI
      WRITE (*,100) PSI
      READ (*,*) RKNVIS
      WRITE (*,110) RKNVIS
      READ (*,*) NGAP
      DO 10 I=1,NGAP
         READ (*,*) SGAP(I),GAPW(I)
         WRITE (*,120) SGAP(I),GAPW(I)
   10 CONTINUE
C
C     FIND DIAMETER OF VENT (GAP)
   50 DO 60 I=1,NGAP
         IF (XPISTON.EQ.SGAP(I)) THEN
            GAP=GAPW(I)
         ELSE
            GAP=GAPW(I-1)+(GAPW(I)-GAPW(I-1))*((XPISTON-SGAP(I-1))/
     +                 (SGAP(I)-SGAP(I-1)))
         ENDIF
   60 CONTINUE
C
C     REYNOLDS NUMBER COMPUTED AND ABS VALUE TAKEN
      RE=ABS(VELH*2.*GAP/RKNVIS)
C
C     COMPUTE LIQUID LOSS
      IF (RE.EQ.0.0) THEN
           RLIQLOS=1.+(1./PSI-1.)*(1./PSI-1.)
           GOTO 200
      ELSE
           RLIQLOS=1.+(1./PSI-1.)*(1./PSI-1.)
     +                +(.3164*RLVENT)/((RE**.25)*2.*GAP)
      ENDIF
```

```
C
  100 FORMAT (' PSI--INLET LOSS          =',F12.5)
  110 FORMAT (' KINEMATIC VISCOSITY      =',F12.5)
  120 FORMAT (' POSITION=',F12.5,10X,' GAP=',F12.5)
  200 RETURN
      END
C     *****************************************************
      SUBROUTINE FRIC2(IOPT,RPISLOS)
C
C     FRICTION LOSS ASSOCIATED WITH PISTON
      COMMON /FI7/ IFRP,NOPT,NFLOSS,FRICPOS(20),FLOSS(20),NFIT
      COMMON /FI9/ PRESRES,VELH,UPISTON,XPISTON
C
C     IOPT=0 ON INPUT; IOPT=1 ON OUTPUT
      IF (IOPT.EQ.1) GOTO 500
C     IF NOPT=1 USE A TABLE OF PISTON POS VS FRICTION LOSS
C     IF NOPT=2 COMPUTE FRICTION AS FCN OF PISTON VELOCITY
      READ(*,*) NOPT
C
C     CHECKING VALUE OF NOPT
      IF ((NOPT.NE.1).AND.(NOPT.NE.2)) THEN
           WRITE(*,5)
    5 FORMAT(' ERROR IN NOPT FROM FRIC2')
           STOP
      ENDIF
C
      GOTO (100,200) NOPT
C
  100 READ (*,*) NFLOSS
      WRITE (*,20) NFLOSS
      DO 10 I=1,NFLOSS
          READ (*,*) FRICPOS(I),FLOSS(I)
          WRITE (*,30) FRICPOS(I),FLOSS(I)
   10 CONTINUE
      READ (*,*) NFIT
      WRITE (*,40) NFIT
C
      GOTO 800
C
C     IF NOPT=2 WILL USE FRIC=(OPPOSITE SIGN OF UPISTON)*B*(V**N)
C     CALL B THE COEF AND N THE EXPONENT
  200 READ (*,*) COEF,EXP
      WRITE (*,50) COEF
      WRITE (*,60) EXP
C
      GOTO 800
C
C     NOPT=1 USES TABLE; NOPT=2 USES FCN OF PISTON VEL
  500 IF (NOPT.EQ.2) GOTO 600
C
C     DETERMINE FRICTION LOSS FROM INTERPOLATED VALUE
C
      IF (XPISTON.GE.FRICPOS(NFLOSS)) THEN
           RPISLOS=FLOSS(NFLOSS)
      ELSE
           CALL DVDINT(XPISTON,RPISLOS,FRICPOS,FLOSS,NFLOSS,NFIT)
      ENDIF
C
      GOTO 800
C
C     DETERMINE FRICITON LOSS FROM RELATION
C         FRICTION=(OPPOSITE SIGN OF PISTON VEL)*B*UPISTON^N
C
  600 IF (UPISTON.EQ.0.0) THEN
           RPISLOS=0.0
           GOTO 800
      ENDIF
      SIGN=-1.*(UPISTON/(ABS(UPISTON)))
      RPISLOS=SIGN*COEF*((ABS(UPISTON))**EXP)
C
   20 FORMAT (' NO. OF PTS FOR FRIC LOSS=',I5)
   30 FORMAT (' PISTON POSITION=',F12.5,'  FRICTION LOSS=',E12.5)
   40 FORMAT (' NO. OF PTS. USED FOR INTERPOLATION:',I3)
   50 FORMAT (' COEF OF PIS FRIC          =',E12.5)
   60 FORMAT (' EXP OF PIS FRIC           =',F12.5)
  800 RETURN
      END
C     *****************************************************
      SUBROUTINE INITIAL
C     SETS INITIAL CONDITIONS
      COMMON /FI9/ PRESRES,VELH,UPISTON,XPISTON
      READ (*,*) PRESRES,VELH,UPISTON,XPISTON
      WRITE (*,10) PRESRES
      WRITE (*,20) VELH
      WRITE (*,30) UPISTON
      WRITE (*,40) XPISTON
C     CONVERT FROM MPA TO CGS SYSTEM
      PRESRES=PRESRES*1.0E7
   10 FORMAT (' INITIAL PR IN RESERVOIR =',F12.5)
   20 FORMAT (' INITIAL VEL IN VENT     =',F12.5)
   30 FORMAT (' INITIAL PISTON VELOCITY =',F12.5)
   40 FORMAT (' INITIAL PISTON POSITION =',F12.5)
      RETURN
      END
C     *****************************************************
      SUBROUTINE NUMERIC
C     PARAMETERS FOR INTEGRATOR
C     INTEGRATOR  DGEAR FROM IMSL
      COMMON /F10/ TINC,EPS,TOUT
      COMMON /F11/ METH,MITER,KWRITE
C
      READ (*,*) TINC,EPS
      READ (*,*) METH,MITER,KWRITE
      WRITE (*,10) TINC
      WRITE (*,20) EPS
      WRITE (*,30) METH
      WRITE (*,40) MITER
      WRITE (*,50) KWRITE
C
   10 FORMAT (' INTEGRATOR--TINC        =',F12.5)
   20 FORMAT (' INTEGRATOR--EPS         =',F12.5)
   30 FORMAT (' INTEGRATOR--METH        =',I6)
   40 FORMAT (' INTEGRATOR--MITER       =',I6)
   50 FORMAT (' INTEGRATOR--KWRITE      =',I6)
      RETURN
      END
C     ****************************************************
      SUBROUTINE STARTUP (Y,YDOT,N)

C     SETS INITIAL MATRIX Y FOR INTEGRATOR
C     SETS INITIAL YDOT
      DIMENSION Y(4),YDOT(4)
C
      COMMON /FI6/ RHOL,RK1,RK2,VOLFO,PSI,AV
      COMMON /FI9/ PRESRES,VELH,UPISTON,XPISTON
      COMMON /F12/ TMS,RMASSL,PRESRESN,PRESCH,RHOLN,BRPRES
      COMMON /F22/ CUPISTON,CVELH,CXPISTON,CRMASSL
C
C     LOADING ARRAY Y
C     Y(1) IS VELOCITY OF PISTON
      Y(1)=UPISTON
C     Y(2) IS VELOCITY OF LIQUID IN VENT
      Y(2)=VELH
C     Y(3) IS POSITION OF PISTON
      Y(3)=XPISTON
C     Y(4) IS MASS OF LIQUID
      RMASSL=RHOL*VOLFO
      Y(4)=RMASSL
      N=4
C
      CUPISTON=0.0
      CVELH=0.0
      CXPISTON=0.0
      CRMASSL=0.0
      YDOT(1)=CUPISTON
      YDOT(2)=CVELH
      YDOT(3)=CXPISTON
      YDOT(4)=CRMASSL

      RETURN
      END
C     ****************************************************
      SUBROUTINE F(N,TIME,Y,YDOT)
C
C     CALLS THE SET OF DIFFERENTIAL EQUATIONS UNDER CONSIDERATION
C
      DIMENSION Y(4),YDOT(4)
      COMMON /NAMES/ TITLE,DATA,GRAF,GEOM,VPDIST
      COMMON /FI2/ AC,AP,AR,AL,IDIFEQN
      COMMON /FI3/ RLPRIME,RLVENT,RLMAX,RLTEMP,RLT
      COMMON /FI4/ PMASS,OFFSET
      COMMON /FI5/ TVENT,NROD,SROD(20),RROD(20),AROD(20),JFLAG
      COMMON /FI6/ RHOL,RK1,RK2,VOLFO,PSI,AV
      COMMON /FI7/ IFRP,NOPT,NFLOSS,FRICPOS(20),FLOSS(20),NFIT
      COMMON /FI8/ TI(500),CCP(500),FLAG
      COMMON /FI9/ PRESRES,VELH,UPISTON,XPISTON
      COMMON /F10/ TINC,EPS,TOUT
      COMMON /F11/ METH,MITER,KWRITE
      COMMON /F12/ TMS,RMASSL,PRESRESN,PRESCH,RHOLN,BRPRES
      COMMON /F13/ NPTS,IWRITE
      COMMON /F14/ CD,CDC,RKNVIS
      COMMON /F15/ NBFLAG,NPFLAG,NPIST,SPIST(20),RPIST(20),APIST(20)
      COMMON /F16/ XPIST(1000),VOLF(1000),AVT(1000),ALIQ(1000),NGPTS
      COMMON /F17/ XPISTC,IXPISTC,RLPRIMEO
      COMMON /F18/ ISET,RM,RINTVS12
      COMMON /F19/ RP1,RP2,RP3,RB1,RB2,RB3,VOL12,VOL23
      COMMON /F20/ IFRL,RE,NGAP,SGAP(50),GAPW(50)
      COMMON /F21/ RLBLTF,RLBLTS,NPRPTS,ORIGS3,ORIGS2,ORIGS1,X3,X2,X1,IP
      COMMON /F22/ CUPISTON,CVELH,CXPISTON,CRMASSL
      COMMON /F23/ IPRES,RADDP,RADDB,RATIOP,RATIOB
      COMMON /F24/ S3,S2,S1,RINTAR,RINTP3
C
      CHARACTER*80 TITLE
      CHARACTER*80 DATA
      CHARACTER*80 GEOM
      CHARACTER*80 GRAF
      CHARACTER*80 VPDIST
      INTEGER TVENT,FLAG
       IF (IDIFEQN.EQ.1) CALL DIFFEQN1(N,TIME,Y,YDOT)
      RETURN
      END
C     ******************************************************
      SUBROUTINE PRESCCH(IOPT)
C
C     CREATES TIME--COMB CH PRESSURE BOUNDARY CONDITION
C     FROM THE DATA FILE LISTED IN THE BATCH RUN
C
      COMMON /NAMES/ TITLE,DATA,GRAF,GEOM,VPDIST
      COMMON /FI8/ TI(500),CCP(500),FLAG
      COMMON /F10/ TINC,EPS,TOUT
      COMMON /F12/ TMS,RMASSL,PRESRESN,PRESCH,RHOLN,BRPRES
      COMMON /F15/ NBFLAG,NPFLAG,NPIST,SPIST(20),RPIST(20),APIST(20)
C
      CHARACTER*80 TITLE
      CHARACTER*80 DATA
      CHARACTER*80 GEOM
      CHARACTER*80 GRAF
      CHARACTER*80 VPDIST
      CHARACTER*80 IDENT
      INTEGER FLAG
C
C     ARRAY IS FILLED FROM DATA FILE
C     IF IOPT=2 C CH PRES IS INTERPOLATED FROM TABLE
C
      IF (IOPT.EQ.2) GOTO 100
C
C     FILLING ARRAY FROM DATA FILE
      OPEN (16,FILE=DATA,STATUS='OLD',IOSTAT=IOS,ERR=10)
      REWIND(16)
   10 IF (IOS.NE.0) WRITE (6,20) IOS
   20 FORMAT (' ERROR OPENING TIME-C CH PRES FILE',I10)
      READ (16,30) IDENT
      WRITE (*,40) IDENT
      READ (16,*) IPOINTS
      DO 50 I=1,IPOINTS
         READ (16,*,ERR=60,END=70) TI(I),CCP(I)
C
C     CHANGE TIME TO SECONDS
C     CHANGE PRESSURE FROM MPA TO CONSISTENT UNITS WITH CGS SYSTEM
C     BY MULTIPLYING BY 1.0E7 CONVERSION CONSTANT
         TI(I)=TI(I)*1.0E-3
         CCP(I)=CCP(I)*1.0E7
   50 CONTINUE
      GOTO 70
   30 FORMAT (A)
   40 FORMAT (//,' TIME-PRES IDENT:',A,//)
   60 WRITE (6,80)
   80 FORMAT (' ERROR READING TIME-C CH PRES FILE')
   70 CLOSE (16)
      GOTO 400
C
C     INTERPOLATING TO FIND C CH PRES
  100 IF (NPFLAG.EQ.1) THEN
         TMS1=0.0
         NPFLAG=2
      ELSE
         TMS1=TMS*1.0E-3+TINC
      ENDIF
      DO 200 I=FLAG,500
         IF (TMS1.LE.TI(I)) GOTO 300
  200 CONTINUE
  300 IF (TMS1.EQ.TI(I)) THEN
         FLAG=I
         PRESCH=CCP(I)
      ELSE
         FLAG=I-1
         PRESCH=CCP(I-1)+(CCP(I)-CCP(I-1))*
     +             ((TMS1-TI(I-1))/(TI(I)-TI(I-1)))
      ENDIF
C
  400 RETURN
      END
C     ***************************************************
      SUBROUTINE DVDINT(X,FX,XT,FT,NP,ND)
C
C     INTERPOLATES A VALUE OF Y AS A FUNCTION OF X
C     USING ANY ORDER OF INTERPOLATION
C     FROM LIBRARY
C     X: THE SENT VALUE OF X
C     FX: THE INTERPOLBTED VALUE OF Y TO BE RETURNED
C     XT: AN ARRAY OF X VALUES
C     FT: A CORRESPONDING ARRAY OF Y VALUES
C     NP: NUMBER OF POINTS IN THE ARRAYS
C     ND: NUMBER OF POINTS TO BE USED FOR THE INTERPOLATING POLYNOMIAL
C         TWO PTS FOR LINEAR, THREE FOR QUADRATIC, ETC.
C
C     CAUTION: CHECK TO SEE IF THE VALUE OF X IS OUTSIDE THE ARRAY
C              BEFORE ENTERING THE SUBROUTINE
C
      DIMENSION XT(NP),FT(NP),T(16)
      N=ND
 31   N1=(N-1)/2
      N2=N/2
      N3=NP-N2+1
      IF(NP-N)30,41,41
 41   N4=N1+2
      IF(XT(1)-XT(2))22,80,60
22    CONTINUE
      IF(X-2.*XT(1)+XT(2))20,20,21
21    IF(X-2.*XT(NP)+XT(NP-1))441,441,20
441   IF(NP.LT.10)GO TO 42
      N5=NP-N
443   N5=N5/2
      N6=N4+N5
      IF(XT(N6).LT.X)N4=N6
      IF(N5.GT.1)GO TO 443
 42   IF(X-XT(N4))45,43,43
 43   IF(N4-N3)44,45,44
 44   N4=N4+1
      GOTO 42
 45   N4=N4-1
      N5=N4-N1
      DO46I=1,N
      T(I)=FT(N5)
 46   N5=N5+1
      L=(N+1)/2
      TR=T(L)
      N6=N4
      N7=N4+1
      JU=1
      N2=N-1
      UN=1.0
      DO12J=1,N2
      N5=N4-N1
      N3=N-J
      DO9I=1,N3
      N8=N5+J
      T(I)=(T(I+1)-T(I))/(XT(N8)-XT(N5))
 9    N5=N5+1
      GOTO(10,11),JU
      CALL GOTOER
10    UN=UN*(X-XT(N6))
      JU=2
      N6=N6-1
      GOTO 12
11    UN=UN*(X-XT(N7))
      JU=1
      N7=N7+1
      L=L-1
 12   TR=TR+UN*T(L)
      FX=TR
      RETURN
 20   WRITE (*,50) X,XT(1),XT(NP)
      STOP
 50       FORMAT(' ARG. NOT IN TABLE  X=',E14.7,' XT(1)=',
     1    E14.7,' XT(NP)=',E14.7,2X,' DVDINT')
 30   WRITE (*,51) NP,ND
 51   FORMAT(' TABLE TOO SMALL  NP=',I5,' ND= ',I5,2X,' DVDINT')
      STOP
60    IF(X-2.*XT(1)+XT(2))61,20,20
61    IF(X-2.*XT(NP)+XT(NP-1))20.721,721
721   IF(NP.LT.10)GO TO 72
      N5=NP-N
723   N5=N5/2
      N6=N4+N5
      IF(XT(N6).GT.X) N4=N6
      IF(N5.GT.1) GO TO 723
72    IF(X-XT(N4)) 73,73,45
73    IF(N4-N3) 74,45,74
74    N4=N4+1
      GOTO 72
80    WRITE (*,52) XT(1)
      STOP
   52 FORMAT(' CONSTANT TABLE XT(1)=',E14.7,2X,' DVDINT')
      END
C     **************************************************
      SUBROUTINE GOTOER
C
      WRITE(*,10)
   10 FORMAT(/,' ERROR IN COMPUTED GOTO--TERMINATING')
      STOP
      END
C     **************************************************
      SUBROUTINE CAPTION(IDIFEQN)
C     CAPTION ON OUTPUT TABLE
      IF ((IDIFEQN.EQ.1).OR.(IDIFEQN.EQ.2)) THEN
         WRITE (*,10)
   10 FORMAT(1X,'TIME',5X,'CH PR',2X,2X,'LIQ PR',2X,2X,'BRCH PR',1X,
     +2X,'LIQ VEL',
     +1X,4X,'AV',4X,4X,'RHO',3X,1X,'LIQ VOL',1X,1X,'LIQ MASS',1X,
     +2X,'PIST POS',1X,1X,'PIST VEL',1X,4X,'CD')
         WRITE(*,20)
   20 FORMAT(/,1X,'(MS)',5X,'(MPA)',2X,3X,'(MPA)',2X,3X,'(MPA)',2X,
     +2X,'(CM/S)',2X,
     +2X,'(CM**2)',2X,1X,'(G/CC)',2X,3X,'(CC)',3X,3X,'(G)',4X,
     +3X,'(CM)',3X,2X,'(CM/S)',2X,/)
      ELSE IF (IDIFEQN.EQ.3) THEN
         WRITE (*,30)
   30 FORMAT(1X,'TIME',5X,'CH PR',2X,2X,'LIQ PR',2X,2X,'LIQ VEL',
     +1X,4X,'AV',4X,4X,'RHO',3X,1X,'LIQ VOL',1X,1X,'LIQ MASS',1X,
     +2X,'PIST POS',1X,1X,'PIST VEL',1X,4X,'CD',4X,4X,'CDC')
         WRITE(*,40)
   40 FORMAT(/,1X,'(MS)',5X,'(MPA)',2X,3X,'(MPA)',2X,2X,'(CM/S)',2X,
     +2X,'(CM**2)',2X,1X,'(G/CC)',2X,3X,'(CC)',3X,3X,'(G)',4X,
     +3X,'(CM)',3X,2X,'(CM/S)',2X,/)
      ENDIF
C
      RETURN
      END
C     **************************************************
      SUBROUTINE OUT (A,ADOT)
C     OUTPUT TABLE OF RESULTS
```

```
C
      DIMENSION A(4),ADOT(4)
      COMMON /NAMES/ TITLE,DATA,GRAF,GEOM,VPDIST
      COMMON /FI2/ AC,AP,AR,AL,IDIFEQN
      COMMON /FI3/ RLPRIME,RLVENT,RLMAX,RLTEMP,RLT
      COMMON /FI4/ PMASS,OFFSET
      COMMON /FI5/ TVENT,NROD,SROD(20),RROD(20),AROD(20),JFLAG
      COMMON /FI6/ RHOL,RK1,RK2,VOLFO,PSI,AV
      COMMON /FI7/ IFRP,NOPT,NFLOSS,FRICPOS(20),FLOSS(20),NFIT
      COMMON /FI8/ TI(500),BCP(500),FLAG
      COMMON /FI9/ PRESRES,VELH,UPISTON,XPISTON
      COMMON /F10/ TINC,EPS,TOUT
      COMMON /F11/ METH,MITER,KWRITE
      COMMON /F12/ TMS,RMASSL,PRESRESN,PRESCH,RHOLN,BRPRES
      COMMON /F13/ NPTS,IWRITE
      COMMON /F14/ CD,CDC,RKNVIS
      COMMON /F15/ NBFLAG,NPFLAG,NPIST,SPIST(20),RPIST(20),APIST(20)
      COMMON /F16/ XPIST(1000),VOLF(1000),AVT(1000),ALIQ(1000),NGPTS
      COMMON /F17/ XPISTC,IXPISTC,RLPRIMEO
      COMMON /F18/ ISET,RM,RINTVS12
      COMMON /F19/ RP1,RP2,RP3,RB1,RB2,RB3,VOL12,VOL23
      COMMON /F20/ IFRL,RE,NGAP,SGAP(50),GAPW(50)
      COMMON /F21/ RLBLTF,RLBLTS,NPRPTS,ORIGS3,ORIGS2,ORIGS1,X3,X2,X1,IP
      COMMON /F22/ CUPISTON,CVELH,CXPISTON,CRMASSL
      COMMON /F23/ IPRES,RADDP,RADDB,RATIOP,RATIOB
      COMMON /F24/ S3,S2,S1,RINTAR,RINTP3
C
      CHARACTER*80 TITLE
      CHARACTER*80 DATA
      CHARACTER*80 GEOM
      CHARACTER*80 GRAF
      CHARACTER*80 VPDIST
      INTEGER TVENT,FLAG
C
C     CALL TO EQUATIONS TO SET ALL VARIABLES AT CURRENT PISTON
C     VELOCITY, LIQ VEL, PISTON POSITION, LIQ MASS
C     ON FIRST STEP WRITE INITIAL CONDITIONS
      IF ((TMS.GT.0.1E-7).AND.(IDIFEQN.EQ.1))
     +          CALL DIFFEQN1(N,TIME,A,ADOT)
C
C     CONVERTING PRESSURE TO MPA
      PRESRESN=PRESRESN*(1.0E-7)
      PRESCH=PRESCH*(1.0E-7)
      BRPRES=BRPRES*(1.0E-7)
C
C     WRITING TO PRINTER
      IF ((IDIFEQN.EQ.1).OR.(IDIFEQN.EQ.2)) THEN
         WRITE(*,10) TMS,PRESCH,PRESRESN,BRPRES,A(2),AV,
     +          RHOLN,VOLFO,A(4),A(3),A(1),CD
   10       FORMAT(F5.2,11F10.3)

         NPTS=NPTS+1
         ENDFILE (6)
         CALL GRAFFILE(A,ADOT)
C
      ELSE IF (IDIFEQN.EQ.3) THEN
```

```
C     INCLUDE COMPARISON DC=V3/SQRT(2*(P1-P3)/RHO)
        WRITE(*,20) TMS,PRESCH,PRESRESN,A(2),AV,RHOLN,VOLFO,
     +            A(4),A(3),A(1),CD,CDC
   20          FORMAT(F5.2,11F10.3)
        NPTS=NPTS+1
        CALL GRAFFILE(A,ADOT)
      ENDIF
C
C     CALL PRESSURE DISTRIBUTION OUTPUT IF IPRES=1
      IF (IPRES.EQ.1) THEN
C     CONVERT PRESSURE BACK TO CGS SYSTEM
        PRESRESN=PRESRESN*(1.0E+7)
        PRESCH=PRESCH*(1.0E+7)
        BRPRES=BRPRES*(1.0E+7)
        CALL PRESDIS(2)
      ENDIF
C
C     DIAGNOSTICS IF IWRITE=1
      IF (IWRITE.EQ.1) THEN
C
      UPISTON=A(1)
      VELH=A(2)
      XPISTON=A(3)
      RMASSL=A(4)
C
C     SLOPE OF PISTON BACK FACE
      RM=(RP1-RP2)/RLPRIME
C
C     LENGTHS OF RESERVOIR AT TIME TMS
      RL13=RLPRIME+RLVENT
      RL03=RLT+RL13
C
C     VOLUME IN REGION 1-2 DECREASES WHEN RLT=0.0
      IF (RLT.EQ.0.0) VOL12=VOLFO-VOL23
      VOL13=VOL12+VOL23
C
      WRITE(*,90) CUPISTON,CVELH,CXPISTON,CRMASSL
   90 FORMAT(' CUPISTON=',G12.5,'   CVELH=',G12.5,'   CXPISTON=',
     +  G12.5,'   CRMASSL=',G12.5)
C
      ENDIF
      RETURN
      END
C     ****************************************************************
      SUBROUTINE GRAFFILE(A,ADOT)
C     CREATES FILE TO USE FOR GRAPHING
C     ORDER OF VARIABLES SAME AS IN SUBROUTINE OUT
C
      DIMENSION A(4),ADOT(4)
      COMMON /NAMES/ TITLE,DATA,GRAF,GEOM,VPDIST
      COMMON /FI2/ AC,AP,AR,AL,IDIFEQN
      COMMON /FI5/ TVENT,NROD,SROD(20),RROD(20),AROD(20),JFLAG
      COMMON /FI6/ RHOL,RK1,RK2,VOLFO,PSI,AV
      COMMON /FI9/ PRESRES,VELH,UPISTON,XPISTON
      COMMON /F12/ TMS,RMASSL,PRESRESN,PRESCH,RHOLN,BRPRES
      COMMON /F13/ NPTS,IWRITE
      COMMON /F14/ CD,CDC,RKNVIS
C
      CHARACTER*80 TITLE
      CHARACTER*80 DATA
      CHARACTER*80 GEOM
      CHARACTER*80 GRAF
      CHARACTER*80 VPDIST
      INTEGER TVENT
C
      IF (NPTS.EQ.1) OPEN(17,FILE=GRAF,IOSTAT=IOS,ERR=10)
C
      IF ((IDIFEQN.EQ.1).OR.(IDIFEQN.EQ.2)) THEN
         WRITE(17,30) TMS,PRESCH,PRESRESN,BRPRES,A(2),AV,
     +             RHOLN,VOLFO,A(4),A(3),A(1),CD
         ENDFILE (17)
      ELSE IF (IDIFEQN.EQ.3) THEN
         WRITE(17,30) TMS,PRESCH,PRESRESN,A(2),AV,RHOLN,VOLFO,
     +             A(4),A(3),A(1),CD,CDC
      ENDIF
      GOTO 40
C
   10 IF (IOS.NE.0) WRITE (6,20) IOS
   20 FORMAT(' ERROR OPENING FILE FOR GRAPH DATA',I10)
   30 FORMAT(F5.2,11F10.3)
C
   40 RETURN
      END
C     ******************************************************
      SUBROUTINE DIFFEQN1(N,TIME,Y,YDOT)
C
C     VALUES OF DERIVATIVES OF PISTON VEL, VEL OF LIQUID IN VENT,
C              POSITION OF PISTON, MASS OF LIQUID
C
      DIMENSION Y(4),YDOT(4)
      COMMON /NAMES/ TITLE,DATA,GRAF,GEOM,VPDIST
      COMMON /FI2/ AC,AP,AR,AL,IDIFEQN
      COMMON /FI3/ RLPRIME,RLVENT,RLMAX,RLTEMP,RLT
      COMMON /FI4/ PMASS,OFFSET
      COMMON /FI5/ TVENT,NROD,SROD(20),RROD(20),AROD(20),JFLAG
      COMMON /FI6/ RHOL,RK1,RK2,VOLFO,PSI,AV
      COMMON /FI7/ IFRP,NOPT,NFLOSS,FRICPOS(20),FLOSS(20),NFIT
      COMMON /FI8/ TI(500),CCP(500),FLAG
      COMMON /FI9/ PRESRES,VELH,UPISTON,XPISTON
      COMMON /F10/ TINC,EPS,TOUT
      COMMON /F11/ METH,MITER,KWRITE
      COMMON /F12/ TMS,RMASSL,PRESRESN,PRESCH,RHOLN,BRPRES
      COMMON /F13/ NPTS,IWRITE
      COMMON /F14/ CD,CDC,RKNVIS
      COMMON /F15/ NBFLAG,NPFLAG,NPIST,SPIST(20),RPIST(20),APIST(20)
      COMMON /F16/ XPIST(1000),VOLF(1000),AVT(1000),ALIQ(1000),NGPTS
      COMMON /F17/ XPISTC,IXPISTC,RLPRIMEO
      COMMON /F18/ ISET,RM,RINTVS12
      COMMON /F19/ RP1,RP2,RP3,RB1,RB2,RB3,VOL12,VOL23
      COMMON /F20/ IFRL,RE,NGAP,SGAP(50),GAPW(50)
      COMMON /F21/ RLBLTF,RLBLTS,NPRPTS,ORIGS3,ORIGS2,ORIGS1,X3,X2,X1,IP
      COMMON /F22/ CUPISTON,CVELH,CXPISTON,CRMASSL
      COMMON /F23/ IPRES,RADDP,RADDB,RATIOP,RATIOB
      COMMON /F24/ S3,S2,S1,RINTAR,RINTP3
C
      CHARACTER*80 TITLE
      CHARACTER*80 DATA
      CHARACTER*80 GEOM
      CHARACTER*80 GRAF
      CHARACTER*80 VPDIST
      INTEGER TVENT,FLAG
C
      PARAMETER (PI=3.141592654)
C
      EXTERNAL FAREA,FAI1A,F1A,FAIVA,FVA,FAIVA2,FXA,FV2A,FVOL,FVA2
C
C     FINAL VALUES OF INTEGRALS: RI---
C     INTERMEDIATE VALUES OF INTEGRALS: RIN---
C     FUNCTIONS TO EVALUATE INTEGRANDS: F-----
C
C     PASSING ARRAY Y TO VARIABLES
C     VELOCITY OF PISTON
      UPISTON=Y(1)
C     VELOCITY OF LIQUID IN VENT
      VELH=Y(2)
C     POSITION OF PISTON
      XPISTON=Y(3)
C     MASS OF LIQUID
      RMASSL=Y(4)
C     VOLUME OF FUEL DEPENDENT UPON PISTON POSITION
C     IF TVENT=1 THEN STRAIGHT BOLT
C     IF TVENT=2 THEN LOOKUP GEOMETRY RECORDED IN FILE
C      FROM BOLTGEO.FOR (C630MM.DAT)
C             //IOPT=2 TO FIND VOLUME,
C            (VOLFO), AREA OF VENT (AV), AREA OF LIQUID (AL),
C            RECOMPUTE LENGTH RLT, RECOMPUTE LENGTH RLPRIME
C
      IF (TVENT.EQ.1) CALL BOLT1(2)
      IF (TVENT.EQ.2) CALL BOLT2(2)
C
C     DENSITY OF LIQUID
      RHOLN=RMASSL/VOLFO
C
C     PRESSURE IN LIQUID RESERVOIR
      PRESRESN=PRESRES+(RK1/RK2)*((RHOLN/RHOL)**RK2-1.)
C
C     PRESSURE IN COMBUSTION CHAMBER IS FOUND BY TABLE
C            LOOKUP OF BOUNDARY CONDITIONS
C            IOPT=2 TO READ FROM TABLE
C
      CALL PRESCCH(2)
C
C     ORIGINAL POSITIONS ON PISTON AND BOLT WITH ZERO
C     AT THE BACK WALL AND POSITIVE TOWARD THE FRONT
C     OF THE BOLT
      ORIGS3=RLMAX+RLVENT
      ORIGS2=ORIGS3-RLVENT
      ORIGS1=ORIGS2-RLPRIME
      X3=ORIGS3+OFFSET
      X2=X3-RLBLTF
      X1=X2-RLBLTS
C
C     DIFFERENCE IN RADII IN PISTON AND BOLT SLANTS
      RADDP=RP2-RP1
      RADDB=RB2-RB1
C
C     RATIOS GIVING SLOPE OF SLANT SECTIONS
      RATIOP=RADDP/RLPRIME
      RATIOB=RADDB/RLBLTS
C
C     RLMAX IS MAX PISTON DISP AND DOES NOT INCLUDE VENT
C     SETTING CURRENT LOCATIONS OF POINTS ON PISTON
      S3=ORIGS3-XPISTON
      S2=ORIGS2-XPISTON
      S1=ORIGS1-XPISTON
C
C     AREA OF BACK WALL
      AW=PI*(RP1*RP1-RB1*RB1)
C
C     FORCE ON COMBUSTION CHAMBER SIDE MUST BE GREATER
C     THAN FORCE ON LIQUID RESERVOIR SIDE
C     BEFORE PISTON CAN MOVE
      IF (PRESCH.LE.45692100.0) THEN
         CUPISTON=0.0
         CVELH=0.0
         CXPISTON=0.0
         CRMASSL=0.0
         CVELSP=0.0
         CPOSSP=0.0
         BRPRES=PRESRESN
         GOTO 300
      ENDIF
C
C     TIME DERIVATIVE OF VENT AREA
      IF ((X1.LT.S3).AND.(S3.LT.X2)) THEN
         CAV=2.*PI*BLTRAD(S3)*UPISTON*RATIOB
      ELSE
         CAV=0.
      ENDIF
C
C     INTEGRAL K1
C     INTEGRAL FROM 0 TO S3 1./A(X,T)
         CALL QROMO(F1A,S1,S3,RIN1A)
         RI1A=S1/AW+RIN1A
C
C     INTEGRAL K2
C     INTEGRAL FROM 0 TO S3 OF A(X,T) TIMES INTEGRAL FROM
C     0 TO X OF 1./A(X,T)
C     BOOK 5, P 81 (38,39)
      IF (X1.LT.S3) THEN
         CALL QROMO(FAREA,S1,S3,RINA)
         CALL QROMO(FAI1A,S1,S3,RINA1A)
         RIDA1A=S1*S1/2.+S1*RINA/AW+RINA1A
      ELSE IF (S3.LE.X1) THEN
```

```
         CALL QROMO(FAREA,S1,S2,RINA)
         CALL QROMO(FAI1A,S1,S2,RINA1A)
         CALL QROMO(F1A,S1,S2,RIN1A)
         RIDA1A=S1*S1/2.+S1*RINA/AW+RINA1A
     +             +AV*S1*(S3-S2)/AW+AV*RIN1A*(S3-S2)+(S3-S2)*(S3-S2)/2.
      ENDIF
C
C     INTEGRAL K3
C     INTEGRAL FROM 0 TO S3 OF V(X,T)*V(X,T)/A(X,T)
         CALL QROMO(FV2A,S1,S3,RINV2A)
         RIV2A=AW*S1*S1*S1/3.+RINV2A
C
C     INTEGRAL K4
C     INTEGRAL FROM 0 TO S3 OF V(X,T)/A(X,T)
         CALL QROMO(FVA,S1,S3,RINVA)
         RIVA=S1*S1/2.+RINVA
C
C     INTEGRAL K5
C     INTEGRAL FROM 0 TO S3 OF X*A(X,T)
         CALL QROMO(FXA,S1,S3,RINXA)
         RIXA=AW*S1*S1/2.+RINXA
C
C     INTEGRAL K6
C     INTEGRAL FROM 0 TO S3 OF A(X,T) TIMES INTEGRAL FROM
C     0 TO X OF V(X,T)/A(X,T)
         CALL QROMO(FAREA,S1,S3,RINA)
         CALL QROMO(FAIVA,S1,S3,RINAVA)
         RIDAVA=AW*S1*S1*S1/6.+S1*S1*RINA/2.+RINAVA
C
C     INTEGRAL K7
C     INTEGRAL FROM 0 TO S3 V(X,T)
         CALL QROMO(FVOL,S1,S3,RINV)
         RIV=S1*S1*AW/2.+RINV
C
C     INTEGRAL K8
C     INTEGRAL FROM 0 TO S3 OF A(X,T) TIMES INTEGRAL FROM
C     0 TO X OF V(X,T)/[A(X,T)*A(X,T)]
         CALL QROMO(FAREA,S1,S3,RINA)
         CALL QROMO(FAIVA2,S1,S3,RIAVA2)
         RIDVA2=S1*S1*S1/6.+S1*S1*RINA/(2.*AW)+RIAVA2
C
C     INTEGRAL K9
C     INTEGRAL FROM 0 TO S3 OF V(X,T)/[A(X,T)*A(X,T)]
         CALL QROMO(FVA2,S1,S3,RINVA2)
         RIVA2=S1*S1/(2.*AW)+RINVA2
C
C     TERMS USED BELOW
      Z1=(VELH*AV-UPISTON*AR)/VOLFO
      Z2=(VELH+CAV+AV*CVELH
     +    -AR*CUPISTON+Z1*UPISTON*(AR+AV))/VOLFO
C
C     COEFFICIENTS OF INTEGRALS
C
      CK1=RHOLN*AW*AW*UPISTON*(UPISTON/(2.*VOLFO))+RHOLN*Z1*AW*UPISTON
      CK2=-1.*RHOLN*Z1*(AR+AV)*(UPISTON/VOLFO)
      CK3=RHOLN*Z1*(Z1/(2.*VOLFO))
      CK4=RHOLN*AW*Z1*(UPISTON/VOLFO)-RHOLN*CAV*(VELH/VOLFO)
     +             -Z1*RHOLN*(AR+AV)*(UPISTON/VOLFO)
      CK5=RHOLN*Z1*(UPISTON/VOLFO)
      CK6=RHOLN*CAV*(VELH/(VOLFO*VOLFO))
     +              +Z1*RHOLN*(AR+AV)*(UPISTON/(VOLFO*VOLFO))
      CK7=-1.*RHOLN*Z1*(UPISTON/VOLFO)
      CK8=-1.*RHOLN*Z1*(UPISTON/VOLFO)
      CK9=RHOLN*UPISTON*Z1
C
      TVA=-1.*RHOLN*UPISTON*AW*UPISTON*AW/(2.*AV*AV)
     +          -RHOLN*VOLFO*VOLFO*Z1*Z1/(2.*AV*AV)
     +          -RHOLN*UPISTON*AW*VOLFO*Z1/(AV*AV)
     +          +RHOLN*UPISTON*VOLFO*Z1/AV
     +          -RHOLN*UPISTON*Z1*S3
C
C     CONTINUITY EQN
      SCKC=CK1*RI1A+CK2*RIDA1A+CK3*RIV2A+CK4*RIVA+CK5*RIXA
     +       +CK6*RIDAVA+CK7*RIV+CK8*RIDVA2+CK9*RIVA2+TVA
C
      CUDOTC=-1.*RHOLN*AW*RI1A+RHOLN*AW*RIDA1A/VOLFO
     +            -RHOLN*RIXA/VOLFO-RHOLN*AR*RIDAVA/(VOLFO*VOLFO)
     +            +RHOLN*S3+RHOLN*AR/VOLFO
C
      CVDOTC=RHOLN*AV*RIDAVA/(VOLFO*VOLFO)-RHOLN*AV*RIVA/VOLFO
C
C     BREECH PRESSURE
      PRX2=(UPISTON*RI1A)*(RHOLN*AW*AW)*(UPISTON/(2.*VOLFO))
      PRX3=(RHOLN*AW*RIDA1A)*(CUPISTON/VOLFO)
      PRX4=.5*RHOLN*UPISTON*UPISTON
      PRX5=(RHOLN*Z1*Z1*RIV2A)/(2.*VOLFO)
      PRX6=(RHOLN*Z1*AW*RIVA)*(UPISTON/VOLFO)
      PRX7=(RHOLN*RIXA)*(CUPISTON/VOLFO)
      PRX8=(RHOLN*Z2*RIDAVA)/VOLFO
      PRX9=(RHOLN*Z1*RIV)*(UPISTON/VOLFO)
      PRX10=(RHOLN*Z1*AW*RIDA1A)*(UPISTON/VOLFO)
      PRX11=(RHOLN*Z1*RIDVA2)*(UPISTON/VOLFO)
      PRX26=(RHOLN*Z1*RIXA)*(UPISTON/VOLFO)
C
      BRPRES=PRESRESN+PRX2+PRX3-PRX4+PRX5+PRX6-PRX7+PRX8-PRX9-PRX10
     +  -PRX11+PRX26
C
C     TERMS IN MOMENTUM EQUATIO
      CMK7=RHOLN*CAV*(VELH/VOLFO)
     +     +RHOLN*(AR+AV)*Z1*(UPISTON/VOLFO)-RHOLN*Z1*Z1
C
C     TIME DERIVATIVE OF K7
      DELK7=-1.*AW*S3*UPISTON
      TMK7=CMK7*RIV+RHOLN*Z1*DELK7
C
C     MOMENTUM EQUATION
      SAM=PRESCH*(AP+AV)-BRPRES*AW+RHOLN*AV*VELH*VELH
     +       +(-1.*AW*RHOLN+RHOLN*(AR+AV)+RHOLN*S3*AW*AR/VOLFO
     +          -RHOLN*AR)*UPISTON*UPISTON
     +       +(-1.*RHOLN*AV*AW*S3/VOLFO+RHOLN*AV)*UPISTON*VELH
     +       +TMK7
C
```

```
      CUDOTM=PMASS-AW*RHOLN*S3+RHOLN*VOLFO+AR*RHOLN*(RIV/VOLFO)
      CVDOTM=AV*RHOLN*(RIV/VOLFO)
C
C     DETERMINE FRICTION LOSSES
C     IF IFRL=1 CONSIDER LIQUID FRICTION
C     IF IFRP=1 CONSIDER PISTON FRICTION FROM INPUT TABLE
C     IF IFRP=2 CONSIDER PISTON FRICTION FROM FR=(-+)B*V^N
      IF (IFRL.EQ.1) CALL FRIC1(1,RLIQLOS)
      IF (IFRP.EQ.1) CALL FRIC2(1,RPISLOS)
C
C     CHANGE IN PISTON VELOCITY
      CUPISTON=(CVDOTM/(CUDOTC*CVDOTM+CUDOTM*CVDOTC))*
     +          ((PRESCH-PRESRESN)-SCKC+SAM*CVDOTC/CVDOTM)
C
C     CHANGE IN LIQUID VEL IN VENT
      CVELH=(CVDOTM/(CUDOTC*CVDOTM+CUDOTM*CVDOTC))*
     +       ((PRESCH-PRESRESN)-SCKC+SAM*CVDOTC/CVDOTM)*
     +       (CUDOTM/CVDOTM)-SAM/CVDOTM
C
C     CHANGE IN POSITION OF PISTON
      CXPISTON=UPISTON
C
C     CHANGE IN MASS OF LIQUID
      CRMASSL=-1.*RHOLN*AV*VELH
C
C     DISCHARGE COEFFICIENT (USING V3=VELH AND BREECH PRES)
      IF (PRESRESN.GT.PRESCH) THEN
         CD=VELH/(SQRT(2*(PRESRESN-PRESCH)/RHOLN))
      ELSE
         CD=0.0
      ENDIF
C
  300 CONTINUE
C     FILLING ARRAY YDOT
C     YDOT(1) IS CHANGE IN PISTON VELOCITY
      YDOT(1)=CUPISTON
C     YDOT(2) IS CHANGE IN LIQUID VEL IN VENT
      YDOT(2)=CVELH
C     YDOT(3) IS CHANGE IN POSITION OF PISTON
      YDOT(3)=CXPISTON
C     YDOT(4) IS CHANGE IN MASS OF LIQUID
      YDOT(4)=CRMASSL
      RETURN
      END
C     *********************************************************
      SUBROUTINE PRESDIS(IOPT)
C
C     COMPUTES VELOCITY AND PRESSURE DISTRIBUTION IN LIQUID RESERVOIR

C     INTEGRATION OF AREA AND VOLUME INTEGRALS USED
C     IN PRESSURE DISTRIBUTION

C     USING ROMBERG INTEGRATION WITH REFINEMENT OF MIDPOINT RULE
C
C     FINAL VALUES OF INTEGRALS: RI---
C     INTERMEDIATE VALUES OF INTEGRALS: RIN---
```

```
C     FUNCTIONS TO EVALUATE INTEGRANDS: F-----
      COMMON /NAMES/ TITLE,DATA,GRAF,GEOM,VPDIST
      COMMON /FI2/ AC,AP,AR,AL,IDIFEQN
      COMMON /FI3/ RLPRIME,RLVENT,RLMAX,RLTEMP,RLT
      COMMON /FI4/ PMASS,OFFSET
      COMMON /FI6/ RHOL,RK1,RK2,VOLFO,PSI,AV
      COMMON /FI9/ PRESRES,VELH,UPISTON,XPISTON
      COMMON /F10/ TINC,EPS,TOUT
      COMMON /F12/ TMS,RMASSL,PRESRESN,PRESCH,RHOLN,BRPRES
      COMMON /F19/ RP1,RP2,RP3,RB1,RB2,RB3,VOL12,VOL23
      COMMON /F21/ RLBLTF,RLBLTS,NPRPTS,ORIGS3,ORIGS2,ORIGS1,X3,X2,X1,IP
      COMMON /F22/ CUPISTON,CVELH,CXPISTON,CRMASSL
      COMMON /F23/ IPRES,RADDP,RADDB,RATIOP,RATIOB
      COMMON /F24/ S3,S2,S1,RINTAR,RINTP3
      COMMON /F25/ I
C
      CHARACTER*80 TITLE
      CHARACTER*80 DATA
      CHARACTER*80 GRAF
      CHARACTER*80 GEOM
      CHARACTER*80 VPDIST
      PARAMETER (PI=3.141592654,NDIV=100)
C
      EXTERNAL FAREA,FAI1A,F1A,FAIVA,FVA,FAIVA2,FXA,FV2A,FVOL,FVA2
C
C     COMPUTES PRESSURE DISTRIBUTION ACROSS LIQUID RESERVOIR
C     USING LAGRANGE ASSUMPTIONS
C
C     READ INPUT IF IOPT=1 AND SET CONSTANTS
C     CONSTANTS SET ON FIRST CALL
      IF (IOPT.EQ.2) GOTO 100
C
C     INPUT BOLT FLAT LENGTH AND SLANT LENGTH
      READ (*,*) RLBLTF,RLBLTS
      WRITE (*,40) RLBLTF,RLBLTS
C
C     INPUT NO. PTS FOR PR DIST AND OPTION TO OPEN GRAPH FILE
      READ (*,*) NPRPTS,IP
      WRITE (*,50) NPRPTS,IP
      IF (IP.EQ.1) THEN
         READ (*,70) VPDIST
         WRITE (*,80) VPDIST
      ENDIF
C
C     ORIGINAL POSITIONS ON PISTON AND BOLT WITH ZERO
C     AT THE BACK WALL AND POSITIVE TOWARD THE FRONT
C     OF THE BOLT
      ORIGS3=RLMAX+RLVENT
      ORIGS2=ORIGS3-RLVENT
      ORIGS1=ORIGS2-RLPRIME
      X3=ORIGS3+OFFSET
      X2=X3-RLBLTF
      X1=X2-RLBLTS
C
C     DIFFERENCE IN RADII IN PISTON AND BOLT SLANTS
      RADDP=RP2-RP1
      RADDB=RB2-RB1
C
C     RATIOS GIVING SLOPE OF SLANT SECTIONS
      RATIOP=RADDP/RLPRIME
      RATIOB=RADDB/RLBLTS
C
      GOTO 300
C
C     SET VALUES OF X FOR WHICH PRESSURE IS EVALUATED
C     RLMAX IS MAX PISTON DISP AND DOES NOT INCLUDE VENT
C     THUS, HAVE TO ADD RLVENT TO OBTAIN PRESSURE IN VENT
C

C     SET VALUES OF X FOR WHICH PRESSURE IS EVALUATED
C     RLMAX IS MAX PISTON DISP AND DOES NOT INCLUDE VENT
C     THUS, HAVE TO ADD RLVENT TO OBTAIN PRESSURE IN VENT
C
C     CONSTANTS AT EACH TIMESTEP
C
C     SETTING CURRENT LOCATIONS OF POINTS ON PISTON
  100 S3=ORIGS3-XPISTON
      S2=ORIGS2-XPISTON
      S1=ORIGS1-XPISTON
C
C     AREA OF BACK WALL
      AW=PI*(RP1*RP1-RB1*RB1)
C
C     TIME DERIVATIVE OF VENT AREA
      IF ((X1.LT.S3).AND.(S3.LT.X2)) THEN
         CAV=2.*PI*BLTRAD(S3)*UPISTON*RATIOB
      ELSE
         CAV=0.
      ENDIF
C
C     TERMS USED BELOW
      Z1=(VELH*AV-UPISTON*AR)/VOLFO
      Z2=(VELH+CAV+AV*CVELH
     +     -AR*CUPISTON+Z1*UPISTON*(AR+AV))/VOLFO
C
C     INTEGRAL FROM 0 TO S3 OF A(X,T) TIMES INTEGRAL FROM
C     0 TO X OF 1./A(X,T)
      IF (X1.LT.S3) THEN
         CALL QROMO(FAREA,S1,S3,RINA)
         CALL QROMO(FAI1A,S1,S3,RINA1A)
         RIDA1A=S1*S1/2.+S1*RINA/AW+RINA1A
      ELSE IF (S3.LE.X1) THEN
         CALL QROMO(FAREA,S1,S2,RINA)
         CALL QROMO(FAI1A,S1,S2,RINA1A)
         CALL QROMO(F1A,S1,S2,RIN1A)
         RIDA1A=S1*S1/2.+S1*RINA/AW+RINA1A
     +              +AV*S1*(S3-S2)/AW+AV*RIN1A*(S3-S2)+(S3-S2)*(S3-S2)/2.
      ENDIF
C
C     INTEGRAL FROM 0 TO S3 OF A(X,T) TIMES INTEGRAL FROM
C     0 TO X OF V(X,T)/A(X,T)
         CALL QROMO(FAREA,S1,S3,RINA)
         CALL QROMO(FAIVA,S1,S3,RINAVA)
         RIDAVA-AW*S1*S1*S1/6.+S1*S1*RINA/2.+RINAVA
C
C     INTEGRAL FROM 0 TO S3 OF V(X,T)/A(X,T)
         CALL QROMO(FVA,S1,S3,RINVA)
         RIVA-S1*S1/2.+RINVA
C
C     INTEGRAL FROM 0 TO S3 OF A(X,T) TIMES INTEGRAL FROM
C     0 TO X OF V(X,T)/[A(X,T)*A(X,T)]
         CALL QROMO(FAREA,S1,S3,RINA)
         RIDVA2--S1*S1*S1/6.+S1*S1*RINA/(2.*AW)+RIAVA2
C
C     INTEGRAL FROM 0 TO S3 OF X*A(X,T)
         CALL QROMO(FXA,S1,S3,RINXA)
         RIXA-AW*S1*S1/2.+RINXA
C
C     INTEGRAL FROM 0 TO S3 OF V(X,T)*V(X,T)/A(X,T)
         CALL QROMO(FV2A,S1,S3,RINV2A)
         RIV2A-AW*S1*S1*S1/3.+RINV2A
C
C     INTEGRAL FROM 0 TO S3 1./A(X,T)
         CALL QROMO(F1A,S1,S3,RIN1A)
         RI1A-S1/AW+RIN1A
C
C     INTEGRAL FROM 0 TO S3 V(X,T)
         CALL QROMO(FVOL,S1,S3,RINV)
         RIV-S1*S1*AW/2.+RINV
C
C     STEP SIZE
      STEP-S3/REAL(NDIV)
C
C     LOOPING
      X-0.0
      NPTS-NDIV+1
      DO 400 I=1,NPTS
      IF (I.EQ.NPTS) X-S3
C     VELOCITY DISTRIBUTION
      VOLATX-FVOL(X)
      AREA-FAREA(X)
      VELATX-UPISTON*(AW-AREA)/AREA+(VELH*AV-UPISTON*AR)
     +     *(1./VOLFO)*(VOLATX/AREA)
C
      IF (X.EQ.0.) THEN
         RIVAX-0.
         RIVA2X-0.
         RI1AX-0.
         GOTO 500
      ENDIF
C
C     INTEGRAL FROM 0 TO X OF V(X,T)/A(X,T)
      IF (X.LE.S1) THEN
         RIVAX-X*X/2.
      ELSE IF (X.GT.S1) THEN
         CALL QROMO(FVA,S1,X,RINVA)
         RIVAX-S1*S1/2.+RINVA
      ENDIF
```

```
C
C     INTEGRAL FROM 0 TO X OF V(X,T)/[A(X,T)*A(X,T)]
      IF (X.LE.S1) THEN
         RIVA2X=X*X/(2.*AW)
      ELSE IF (X.GT.S1) THEN
         CALL QROMO(FVA2,S1,X,RINVA2)
         RIVA2X=S1*S1/(2.*AW)+RINVA2
      ENDIF
C
C     INTEGRAL FROM 0 TO X OF 1./A(X,T)
      IF (X.LE.S1) THEN
         RI1AX=X/AW
      ELSE IF (X.GT.S1) THEN
         CALL QROMO(F1A,S1,X,RIN1AX)
         RI1AX=S1/AW+RIN1AX
      ENDIF
C
C     PRESSURE DISTRIBUTION
  500 PRX2=(UPISTON*RI1A)*(RHOLN*AW*AW)*(UPISTON/(2.*VOLFO))
      PRX3=(RHOLN*AW*RIDA1A)*(CUPISTON/VOLFO)
      PRX5=(RHOLN*Z1*Z1*RIV2A)/(2.*VOLFO)
      PRX6=(RHOLN*Z1*AW*RIVA)*(UPISTON/VOLFO)
      PRX7=(RHOLN*RIXA)*(CUPISTON/VOLFO)
      PRX8=(RHOLN*Z2*RIDAVA)/VOLFO
      PRX9=(RHOLN*Z1*RIV)*(UPISTON/VOLFO)
      PRX10=(RHOLN*Z1*AW*RIDA1A)*(UPISTON/VOLFO)
      PRX11=(RHOLN*Z1*RIDVA2)*(UPISTON/VOLFO)
      PRX12=((RHOLN*AW*AW)/(2.*AREA*AREA))*(UPISTON*UPISTON)
      PRX14=(RHOLN*VOLATX*VOLATX*Z1*Z1)/(2.*AREA*AREA)
      PRX16=((RHOLN*AW*Z1*VOLATX)/(AREA*AREA))*UPISTON
      PRX17=((RHOLN*Z1*VOLATX)/AREA)*UPISTON
      PRX18=(-RHOLN*AW*CUPISTON+RHOLN*Z1*UPISTON*AW)*RI1AX
      PRX21=RHOLN*X*CUPISTON
      PRX22=RHOLN*Z2*RIVAX
      PRX24=RHOLN*Z1*RIVA2X*UPISTON
      PRX25=RHOLN*Z1*X*UPISTON
      PRX26=(RHOLN*Z1*RIXA)*(UPISTON/VOLFO)
C
      PRESXT=PRESRESN+PRX2+PRX3+PRX5+PRX6-PRX7+PRX8-PRX9-PRX10
     +  -PRX11-PRX12-PRX14-PRX16+PRX17+PRX18+PRX21
     +  -PRX22+PRX24-PRX25+PRX26
C
C     CONVERTING PRESSURE TO MPA
      PRESXT=PRESXT*(1.0E-7)
C
C     OPEN FILE FOR OUTPUT
      IF (IP.EQ.1)
     +    CALL GRAFDIS(VPDIST,I,TMS,X,VELATX,PRESXT,AREA,VOLATX)
C
C     PRINTOUT OF LOCATION,VELOCITY,PRESSURE,AREA,VOLUME TO PRINTER
C     PRINTOUT AT BACK WALL AND AT EXIT
          WRITE(*,5)
          WRITE (*,10) X,VELATX,PRESXT,AREA,VOLATX
      ENDIF
      IF (I.EQ.NPTS) WRITE(*,10) X,VELATX,PRESXT,AREA,VOLATX
C
```

```
C     PRINT DISTRIBUTION OF POINTS
      J=NDIV/NPRPTS
      XI=REAL(I)
      XJ=REAL(J)
      IF ((INT(XI-INT(XI/XJ)*XJ)).EQ.0)
     +      WRITE (*,10) X,VELATX,PRESXT,AREA,VOLATX
C
C     INCREMENT A STEP
      X=X+STEP
C
  400 CONTINUE
C
    5 FORMAT (/,18X,'X',19X,'VEL',18X,'PRES',19X,'AREA',18X,'VOL')
   10 FORMAT (10X,F12.5,10X,F12.5,10X,F12.5,10X,F12.5,10X,F12.5)
   30 FORMAT (10X,F12.5,10X,F12.5,10X,F12.5,10X,F12.5,10X,F12.5,/)
   40 FORMAT (' FLAT LEN BOLT=',F12.5,' SLANT LEN BOLT=',F12.5)
   50 FORMAT (' NO. OF PTS FOR DIST=',I6,5X,' IP=',I5)
   60 FORMAT (' PISRAD=',F12.5,10X,' BLTRAD=',F12.5,10X,
     +' LIQAREA=',F12.5)
   70 FORMAT (A)
   80 FORMAT (/,' VEL, PRES DIST DATA FILE: ',A)
C
  300 RETURN
      END
C     ***********************************************
      SUBROUTINE GRAFDIS(VPDIST,I,TMS,X,VELATX,PRESXT,AREA,VOLATX)
C
C     CREATES FILE OF VEL, PRES DISTRIBUTION TO USE FOR GRAPHING
C     ORDER OF VARIABLES SAME AS IN SUBROUTINE PRESDIS
C
C     FILE 19: TIME
C              X  VEL(X)  PRES(X)  AREA(X)  VOL(X)
C     FILE 20: SPECIFIC TIMES TO SIMPLIFY GRAPHING
C
      CHARACTER*80 VPDIST
C
C     OPEN FILE ON FIRST CALL
      IF ((TMS.EQ.0.0).AND.(I.EQ.1))
     +        OPEN (19,FILE=VPDIST,IOSTAT=IOS,ERR=10)
C
C     WRITE   LINE 1: TIME FOR EACH NEW TIME
C             LINE 2: TIME, NO. OF COL (USE FOR DISSPLA)
      IF (I.EQ.1) THEN
          WRITE(19,30) TMS
          WRITE(19,35) TMS
   35     FORMAT(F20.8,5X,' 5')
      ENDIF
C
   10 IF (IOS.NE.0.0) WRITE (6,20) IOS
   20 FORMAT(' ERROR OPENING DIST FILE:',I10)
   30 FORMAT(/,' TIME=',F11.5)
   40 FORMAT (F12.5,3X,F12.5,3X,F12.5,3X,F12.5,3X,F12.5)
C
      RETURN
      END
C     *****************************************************
```

```
      FUNCTION IFHEAV(Y,X)
C
C     HEAVISIDE FUNCTION ACTING AS A SWITCH
C
      SWITCH=Y-X
      IF (SWITCH.GE.0) THEN
           IFHEAV=1
      ELSE
           IFHEAV=0
      ENDIF
C
      RETURN
      END
C     ***********************************************
      FUNCTION PISRAD(X)
C
C     FINDS RADIUS OF PISTON AT ANY POSITION X AT A GIVEN TIME
C     WHICH FIXES S1,S2,S3 RELATIVE TO BOLT
C
C
      COMMON /FI6/ RHOL,RK1,RK2,VOLFO,PSI,AV
      COMMON /F19/ RP1,RP2,RP3,RB1,RB2,RB3,VOL12,VOL23
      COMMON /F21/ RLBLTF,RLBLTS,NPRPTS,ORIGS3,ORIGS2,ORIGS1,X3,X2,X1,IP
      COMMON /F23/ IPRES,RADDP,RADDB,RATIOP,RATIOB
      COMMON /F24/ S3,S2,S1,RINTAR,RINTP3
C
      PISRAD=(RP1+RATIOP*(X-S1)*(1-IFHEAV(S1,X)))*IFHEAV(S2,X)
     +     +RP2*(1-IFHEAV(S2,X))*IFHEAV(S3,X)
      RETURN
      END
C     ***********************************************
      FUNCTION BLTRAD(X)
C
C     FINDS RADIUS OF BOLT AT ANY POSITION X AT A GIVEN TIME
C     WHICH FIXES S1,S2,S3 RELATIVE TO BOLT
C
      COMMON /FI6/ RHOL,RK1,RK2,VOLFO,PSI,AV
      COMMON /F19/ RP1,RP2,RP3,RB1,RB2,RB3,VOL12,VOL23
      COMMON /F21/ RLBLTF,RLBLTS,NPRPTS,ORIGS3,ORIGS2,ORIGS1,X3,X2,X1,IP
      COMMON /F23/ IPRES,RADDP,RADDB,RATIOP,RATIOB
      COMMON /F24/ S3,S2,S1,RINTAR,RINTP3

      BLTRAD=(RB1+RATIOB*(X-X1)*(1-IFHEAV(X1,X)))*IFHEAV(X2,X)
     +     +RB2*(1-IFHEAV(X2,X))*IFHEAV(X3,X)
      RETURN
      END
C     ***********************************************
      FUNCTION FAREA(X)
C
C     CALLED TO EVALUATE A(X,T)
C
      COMMON /FI6/ RHOL,RK1,RK2,VOLFO,PSI,AV
      COMMON /F19/ RP1,RP2,RP3,RB1,RB2,RB3,VOL12,VOL23
      COMMON /F21/ RLBLTF,RLBLTS,NPRPTS,ORIGS3,ORIGS2,ORIGS1,X3,X2,X1,IP
      COMMON /F23/ IPRES,RADDP,RADDB,RATIOP,RATIOB
      COMMON /F24/ S3,S2,S1,RINTAR,RINTP3
```

```
C
      PARAMETER (PI=3.141592654)
C
C     TERMS USED TO FIND AREA A(X,T)
      TRMRP1=RP1*RP1*IFHEAV(S2,X)
     +        +RP2*RP2*(1-IFHEAV(S2,X))*IFHEAV(S3,X)
      TRMRP2=RP1*RATIOP*(1-IFHEAV(S1,X))*IFHEAV(S2,X)
      TRMRP3=RATIOP*RATIOP*(1-IFHEAV(S1,X))*IFHEAV(S2,X)
      TRMRP4=TRMRP1-2.*TRMRP2*S1+TRMRP3*S1*S1
      TRMRP5=TRMRP2-TRMRP3*S1
      TRMRB1=RB1*RB1*IFHEAV(X2,X)
     +        +RB2*RB2*(1-IFHEAV(X2,X))*IFHEAV(X3,X)
      TRMRB2=RB1*RATIOB*(1-IFHEAV(X1,X))*IFHEAV(X2,X)
      TRMRB3=RATIOB*RATIOB*(1-IFHEAV(X1,X))*IFHEAV(X2,X)
      TRMRB4=TRMRB1-2.*TRMRB2*X1+TRMRB3*X1*X1
      TRMRB5=TRMRB2-TRMRB3*X1
C
C     TERMS IN AREA OF LIQUID EXPRESSED AS A TRINOMIAL
      TRMAL1=TRMRP4-TRMRB4
      TRMAL2=TRMRP5-TRMRB5
      TRMAL3=TRMRP3-TRMRB3
C
C     AREA OF LIQUID BETWEEN PISTON AND BOLT
      FAREA=PI*(TRMAL1+2.*TRMAL2*X+TRMAL3*X*X)
C
      RETURN
      END
C     ***********************************************
      FUNCTION FVOL(X)
C
C     FINDS VOLUME OF LIQUID AT ANY POSITION X AT A GIVEN TIME
C     WHICH FIXES S1,S2,S3 RELATIVE TO BOLT
C
      COMMON /F16/ RHOL,RK1,RK2,VOLFO,PSI,AV
      COMMON /F19/ RP1,RP2,RP3,RB1,RB2,RB3,VOL12,VOL23
      COMMON /F21/ RLBLTF,RLBLTS,NPRPTS,ORIGS3,ORIGS2,ORIGS1,X3,X2,X1,IP
      COMMON /F23/ IPRES,RADDP,RADDB,RATIOP,RATIOB
      COMMON /F24/ S3,S2,S1,RINTAR,RINTP3
      COMMON /F25/ I
C
      PARAMETER (PI=3.141592654)
C
C     AREA OF BACK WALL
      AW=PI*(RP1*RP1-RB1*RB1)
C
C     VOLUME DEPENDS ON X RELATIVE TO S1,S2,S3 AND X1,X2,X3
      IF (X.LE.S1) VOLLIQ=AW*X
C
C     WITH OFFSET >=.2788 SLANT OF PIST IS NEVER OVER SLANT OF BOLT
      IF ((X.GT.S1).AND.(X.LE.S2)) THEN
         B1=PI*RP1*RP1
         B2=PI*PISRAD(X)*PISRAD(X)
         VOLLIQ=AW*S1+(1./3.)*(X-S1)*(B1+B2+SQRT(B1*B2))
     +         -PI*RB1*RB1*(X-S1)
      ENDIF
C
```

```
      IF (((X.GT.S2).AND.(X.LE.S3)).AND.(X.LE.X1)) THEN
         B1=PI*RP1*RP1
         B2=PI*PISRAD(S2)*PISRAD(S2)
         VOLLIQ=AW*S1+(1./3.)*(S2-S1)*(B1+B2+SQRT(B1*B2))
     +           -PI*RB1*RB1*(S2-S1)
     +           +PI*(RP2*RP2-RB1*RB1)*(X-S2)
      ENDIF
C
      IF (((X.GT.S2).AND.(X.LE.S3)).AND.((X.GT.X1).AND.(X.LE.X2))) THEN
         B1=PI*RP1*RP1
         B2=PI*PISRAD(S2)*PISRAD(S2)
         B3=PI*RB1*RB1
         B4=PI*BLTRAD(X)*BLTRAD(X)
         VOLLIQ=AW*S1+(1./3.)*(S2-S1)*(B1+B2+SQRT(B1*B2))
     +           -PI*RB1*RB1*(S2-S1)+PI*RP2*RP2*(X-S2)
     +           -(1./3.)*(X-S2)*(B3+B4+SQRT(B3*B4))
      ENDIF
C
      IF (((X.GT.S2).AND.(X.LE.S3)).AND.((X.GT.X2).AND.(X.LE.X3)))
     +   VOLLIQ=AW*S1+(1./3.)*(S2-S1)*(B1+B2+SQRT(B1*B2))
     +           -PI*RB1*RB1*(S2-S1)+PI*RP2*RP2*(X2-S2)
     +           -(1./3.)*(X2-S2)*(B3+B4+SQRT(B3*B4))
     +           +(X-X2)*PI*(RP2*RP2-RB2*RB2)
      FVOL=VOLLIQ

      RETURN
      END
C     ************************************************
      FUNCTION F1A(X)
C
C     CALLED TO EVALUATE 1./A(X,T)
      F1A=1./FAREA(X)
      RETURN
      END
C     ************************************************
      FUNCTION FVA(X)
C
C     CALLED TO EVALUATE V(X,T)/A(X,T)
      FVA=FVOL(X)/FAREA(X)
      RETURN
      END
C     ************************************************
      FUNCTION FXA(X)
C
C     CALLED TO EVALUATE X*A(X,T)
      FXA=X*FAREA(X)
      RETURN
      END
C     ************************************************
      FUNCTION FV2A(X)
C
C     CALLED TO EVALUATE V(X,T)*V(X,T)/A(X,T)
      VOL=FVOL(X)
      FV2A=VOL*VOL/FAREA(X)
      RETURN
      END
```

```
C     ************************************************
      FUNCTION FVA2(X)
C
C     CALLED TO EVALUATE V(X,T)/[A(X,T)*A(X,T)]
      AREA-FAREA(X)
      FVA2-FVOL(X)/(AREA*AREA)
      RETURN
      END
C     ************************************************
      FUNCTION FAI1A(X)
C
C     CALLED TO EVALUATE A(X,T)*INTEGRAL 1./A(X,T)
C                                  FROM S1 TO X
      COMMON /F24/ S3,S2,S1,RINTAR,RINTP3
      EXTERNAL F1A
      AREA-FAREA(X)
      CALL QROMO2(F1A,S1,X,RIN1A)
      FAI1A-AREA*RIN1A
      RETURN
      END
C     ************************************************
      FUNCTION FAIVA(X)
C
C     CALLED TO EVALUATE A(X,T)*INTEGRAL V(X,T)/A(X,T)
C                                 FROM S1 TO X
      COMMON /F24/ S3,S2,S1,RINTAR,RINTP3
      EXTERNAL FVA
      AREA-FAREA(X)
      CALL QROMO2(FVA,S1,X,RINVA)
      FAIVA-AREA*RINVA
      RETURN
      END
C     ************************************************
      FUNCTION FAIVA2(X)
C
C     CALLED TO EVALUATE A(X,T)*INTEGRAL V(X,T)/[A(X,T)*A(X,T)]
C                                 FROM S1 TO X
      COMMON /F24/ S3,S2,S1,RINTAR,RINTP3
      EXTERNAL FVA2
      AREA-FAREA(X)
      CALL QROMO2(FVA2,S1,X,RINVA2)
      FAIVA2-AREA*RINVA2
      RETURN
      END
C     ************************************************
C     ---------------> BEGIN INTEGRATOR
C     ADAPTED FROM NUMERICAL RECIPES BY W. PRESS ET AL
C     *******************************************
      SUBROUTINE QROMO(FUNC,A,B,SS)

C     ROMBERG INTEGRATION
      PARAMETER (EPS-1.E-5,JMAX-14,JMAXP-JMAX+1,K-5,KM-K-1)
      DIMENSION S(JMAXP),H(JMAXP)
      EXTERNAL FUNC
      H(1)-1.
      DO 11 J-1,JMAX
```

```
        CALL MIDPNT(FUNC,A,B,S(J),J)
        IF (J.GE.K) THEN
          CALL POLINT(H(J-KM),S(J-KM),K,0.0,SS,DSS)
          IF (ABS(DSS).LT.EPS*ABS(SS)) RETURN
        ENDIF
        S(J+1)=S(J)
        H(J+1)=H(J)/9.
   11    CONTINUE

      WRITE(*,50)
   50 FORMAT(1X,' INTEGRATION CANNOT CONVERGE')

      RETURN
      END
C     *******************************************
      SUBROUTINE MIDPNT(FUNC,A,B,S,N)
      EXTERNAL FUNC
      IF (N.EQ.1) THEN
        Y=0.5*(A+B)
        S=(B-A)*FUNC(Y)
        IT=1
      ELSE
        TNM=IT
        DEL=(B-A)/(3.*TNM)
        DDEL=DEL+DEL
        X=A+0.5*DEL
        SUM=0.
        DO 11 J=1,IT
          SUM=SUM+FUNC(X)
          X=X+DDEL
          SUM=SUM+FUNC(X)
          X=X+DEL
   11      CONTINUE
        S=(S+(B-A)*SUM/TNM)/3.
        IT=3*IT
      ENDIF
      RETURN
      END
C
C     *******************************************
      SUBROUTINE POLINT(XA,YA,N,X,Y,DY)
      PARAMETER (NMAX=10)
      DIMENSION XA(N),YA(N),C(NMAX),D(NMAX)
      NS=1
      DIF=ABS(X-XA(1))

      DO 11 I=1,N
        DIFT=ABS(X-XA(I))
        IF (DIFT.LT.DIF) THEN
          NS=I
          DIF=DIFT
        ENDIF
        C(I)=YA(I)
        D(I)=YA(I)
   11    CONTINUE
      Y=YA(NS)
      NS=NS-1
      DO 13 M=1,N-1
        DO 12 I=1,N-M
          HO=XA(I)-X
          HP=XA(I+M)-X
          W=C(I+1)-D(I)
          DEN=HO-HP
          IF(DEN.EQ.0.)PAUSE
          DEN=W/DEN
          D(I)=HP*DEN
          C(I)=HO*DEN
12         CONTINUE
        IF (2*NS.LT.N-M)THEN
          DY=C(NS+1)
        ELSE
          DY=D(NS)
          NS=NS-1
        ENDIF
        Y=Y+DY
13     CONTINUE
      RETURN
      END
C     *******************************************
C              END INTEGRATOR 1 <==============
C     ===============> BEGIN INTEGRATOR 2
C     *******************************************
      SUBROUTINE QROMO2(FUNC,A,B,SS)

C     ROMBERG INTEGRATION
      PARAMETER (EPS=1.E-5,JMAX=14,JMAXP=JMAX+1,K=5,KM=K-1)
      DIMENSION S(JMAXP),H(JMAXP)
      EXTERNAL FUNC

      H(1)=1.
      DO 11 J=1,JMAX
        CALL MIDPT2(FUNC,A,B,S(J),J)
        IF (J.GE.K) THEN
          CALL POLIN2(H(J-KM),S(J-KM),K,0.0,SS,DSS)
          IF (ABS(DSS).LT.EPS*ABS(SS)) RETURN
        ENDIF
        S(J+1)=S(J)
        H(J+1)=H(J)/9.
11     CONTINUE

      WRITE(*,50)
50    FORMAT(1X,' INTEGRATION CANNOT CONVERGE')

      RETURN
      END
C     *******************************************
      SUBROUTINE MIDPT2(FUNC,A,B,S,N)
      EXTERNAL FUNC
      IF (N.EQ.1) THEN
        Y=0.5*(A+B)
        S=(B-A)*FUNC(Y)
        IT=1
      ELSE
        TNM=IT
        DEL=(B-A)/(3.*TNM)
        DDEL=DEL+DEL
        X=A+0.5*DEL
        SUM=0.
        DO 11 J=1,IT
          SUM=SUM+FUNC(X)
          X=X+DDEL
          SUM=SUM+FUNC(X)
          X=X+DEL
 11        CONTINUE
        S=(S+(B-A)*SUM/TNM)/3.
        IT=3*IT
      ENDIF
      RETURN
      END
C     ***************************************************
      SUBROUTINE POLIN2(XA,YA,N,X,Y,DY)
      PARAMETER (NMAX=10)
      DIMENSION XA(N),YA(N),C(NMAX),D(NMAX)
      NS=1
      DIF=ABS(X-XA(1))

      DO 11 I=1,N
        DIFT=ABS(X-XA(I))
        IF (DIFT.LT.DIF) THEN
          NS=I
          DIF=DIFT
        ENDIF
        C(I)=YA(I)
        D(I)=YA(I)
 11      CONTINUE
      Y=YA(NS)
      NS=NS-1
      DO 13 M=1,N-1
        DO 12 I=1,N-M
          HO=XA(I)-X
          HP=XA(I+M)-X
          W=C(I+1)-D(I)
          DEN=HO-HP
          IF(DEN.EQ.0.)PAUSE
          DEN=W/DEN
          D(I)=HP*DEN
          C(I)=HO*DEN
 12        CONTINUE
        IF (2*NS.LT.N-M)THEN
          DY=C(NS+1)
        ELSE
          DY=D(NS)
          NS=NS-1
        ENDIF
        Y=Y+DY
 13      CONTINUE
      RETURN
      END
```

```
C     *******************************************
C              END INTEGRATOR 2  <------------
C     *******************************************
C     ODE SOLVER--RECEIVED FROM D. KAHANER
C                 NATIONAL BUREAU OF STANDARDS
C     *******************************************
      SUBROUTINE FDUMP
C***BEGIN PROLOGUE  FDUMP
C***DATE WRITTEN   790801   (YYMMDD)
C***REVISION DATE  820801   (YYMMDD)
C***CATEGORY NO.  Z
C***KEYWORDS  ERROR,XERROR PACKAGE
C***AUTHOR  JONES, R. E., (SNLA)
C***PURPOSE  Symbolic dump (should be locally written).
C***DESCRIPTION
C     Written by Ron Jones, with SLATEC Common Math Library Subcommittee
C     Latest revision ---  23 May 1979
C***ROUTINES CALLED  (NONE)
C***END PROLOGUE  FDUMP
C***FIRST EXECUTABLE STATEMENT  FDUMP
      RETURN
      END
      INTEGER FUNCTION I1MACH(I)
C
C   THIS SHORT FUNCTION REPLACES THE ORIGINAL FUNCTION I1MACH(I) BY
C   FOX, HALL AND SCHRYER OF BELL LABS.
C         I-LOK CHANG    JANUARY 6, 1985   IBM PC Family
C
C  imach(1) is standard unit for input
C  imach(2) is standard unit for output
C  imach(4) is standard unit for error messages
C  imach(6) is number of characters per storage unit
C  imach(9) is largest integer
C  imach(10) is base for floating point numbers
C  imach(11) is number of digits in single precision mantissa
C
      INTEGER IMACH(16)
      DATA IMACH/0,6,0,4,0,1,0,0,2147483647,2,23,5*0/
      IF ( (I .EQ. 4) .OR. (I .EQ. 6) .OR. (I.EQ.2)
     *       .OR. (I.EQ.9) .OR. (I.EQ.11) .OR. (I.EQ.10)) GO TO 1
         WRITE(*,5) I
 5       FORMAT(' REQUESTED MACHINE CONSTANT I1MACH(',I2,' )IS NOT
     8AVILABLE.  STOP IN FUNCTION I1MACH(I)')
         STOP
 1    CONTINUE
      I1MACH = IMACH(I)
      RETURN
      END
      INTEGER FUNCTION ISAMAX(N,SX,INCX)
C
C     FIND SMALLEST INDEX OF MAXIMUM MAGNITUDE OF SINGLE PRECISION SX.
C     ISAMAX =  FIRST I, I = 1 TO N, TO MINIMIZE  ABS(SX(1-INCX+I*INCX))
C
      REAL SX(1),SMAX,XMAG
      ISAMAX = 0
      IF(N.LE.0) RETURN
      ISAMAX - 1
      IF(N.LE.1)RETURN
      IF(INCX.EQ.1)GOTO 20
C
C        CODE FOR INCREMENTS NOT EQUAL TO 1.
C
      SMAX - ABS(SX(1))
      NS - N*INCX
      II - 1
          DO 10 I-1,NS,INCX
          XMAG - ABS(SX(I))
          IF(XMAG.LE.SMAX) GO TO 5
          ISAMAX - II
          SMAX - XMAG
    5     II - II + 1
   10     CONTINUE
      RETURN
C
C        CODE FOR INCREMENTS EQUAL TO 1.
C
   20 SMAX - ABS(SX(1))
      DO 30 I - 2,N
         XMAG - ABS(SX(I))
         IF(XMAG.LE.SMAX) GO TO 30
         ISAMAX - I
         SMAX - XMAG
   30 CONTINUE
      RETURN
      END
      FUNCTION J4SAVE(IWHICH,IVALUE,ISET)
C***BEGIN PROLOGUE  J4SAVE
C***REFER TO  XERROR
C     Abstract
C        J4SAVE saves and recalls several global variables needed
C        by the library error handling routines.
C
C     Written by Ron Jones, with SLATEC Common Math Library Subcommittee
C    Adapted from Bell Laboratories PORT Library Error Handler
C     Latest revision ---  23 MAY 1979
C***REFERENCES  JONES R.E., KAHANER D.K., "XERROR, THE SLATEC ERROR-
C                 HANDLING PACKAGE", SAND82-0800, SANDIA LABORATORIES,
C                 1982.
C***ROUTINES CALLED  (NONE)
C***END PROLOGUE  J4SAVE
      LOGICAL ISET
      INTEGER IPARAM(9)
      SAVE IPARAM
      DATA IPARAM(1),IPARAM(2),IPARAM(3),IPARAM(4)/0,2,0,10/
      DATA IPARAM(5)/1/
      DATA IPARAM(6),IPARAM(7),IPARAM(8),IPARAM(9)/0,0,0,0/
C***FIRST EXECUTABLE STATEMENT  J4SAVE
      J4SAVE - IPARAM(IWHICH)
      IF (ISET) IPARAM(IWHICH) - IVALUE
      RETURN
      END
      FUNCTION NUMXER(NERR)
```

```
C***BEGIN PROLOGUE  NUMXER
C***REFER TO  XERROR
C     Abstract
C        NUMXER returns the most recent error number,
C        in both NUMXER and the parameter NERR.
C
C     Written by Ron Jones, with SLATEC Common Math Library Subcommittee
C     Latest revision ---  7 JUNE 1978
C***REFERENCES  JONES R.E., KAHANER D.K., "XERROR, THE SLATEC ERROR-
C                 HANDLING PACKAGE", SAND82-0800, SANDIA LABORATORIES,
C                 1982.
C***ROUTINES CALLED  J4SAVE
C***END PROLOGUE  NUMxER
C***FIRST EXECUTABLE STATEMENT  NUMXER
      NERR = J4SAVE(1,0,.FALSE.)
      NUMXER = NERR
      RETURN
      END
C
      REAL FUNCTION R1MACH(I)
C  this short routine replaces the original function r1mach
C  by fox, hall and schryer.
C
C  i-lok chang   september 17, 1984   For IBM PC family *****
C
C      Changes: (DKK)
C       28 March 1986:
C        Altered rmach(1) from 1.40128e-44 to 1.4e-38
      REAL RMACH(5)
      INTEGER I
      IF ( (I .NE. 1) .AND. (I .NE. 4) .AND. (I.NE.2)) GO TO 1
       RMACH(1) = 1.4E-38
       RMACH(2) = 3.0E+38
       RMACH(4) = 1.192093E-6
       R1MACH = RMACH(I)
      GO TO 5
 1    CONTINUE
       WRITE(*,10) I
 10    FORMAT(' REQUEST INDEX TO R1MACH IS', I3,' PROGRAM STOPS AT
     & FUNCTION R1MACH')
       STOP
 5    CONTINUE
      RETURN
      END
      SUBROUTINE SAXPY(N,SA,SX,INCX,SY,INCY)
C
C     OVERWRITE SINGLE PRECISION SY WITH SINGLE PRECISION SA*SX +SY.
C     FOR I = 0 TO N-1, REPLACE  SY(LY+I*INCY) WITH SA*SX(LX+I*INCX) +
C       SY(LY+I*INCY), WHERE LX = 1 IF INCX .GE. 0, ELSE LX = (-INCX)*N,
C       AND LY IS DEFINED IN A SIMILAR WAY USING INCY.
C
      REAL SX(1),SY(1),SA
      IF(N.LE.0.OR.SA.EQ.0.E0) RETURN
      IF(INCX.EQ.INCY) IF(INCX-1) 5,20,60
    5 CONTINUE
C
```

```
C        CODE FOR NONEQUAL OR NONPOSITIVE INCREMENTS.
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        SY(IY) = SY(IY) + SA*SX(IX)
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      RETURN
C
C        CODE FOR BOTH INCREMENTS EQUAL TO 1
C        CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 4.
C
   20 M = MOD(N,4)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        SY(I) = SY(I) + SA*SX(I)
   30 CONTINUE
      IF( N .LT. 4 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,4
        SY(I) = SY(I) + SA*SX(I)
        SY(I + 1) = SY(I + 1) + SA*SX(I + 1)
        SY(I + 2) = SY(I + 2) + SA*SX(I + 2)
        SY(I + 3) = SY(I + 3) + SA*SX(I + 3)
   50 CONTINUE
      RETURN
C
C        CODE FOR EQUAL, POSITIVE, NONUNIT INCREMENTS.
C
   60 CONTINUE
      NS = N*INCX
          DO 70 I=1,NS,INCX
          SY(I) = SA*SX(I) + SY(I)
   70     CONTINUE
      RETURN
      END
      SUBROUTINE SDCOR (DFDY,EL,FA,H,IMPL,IPVT,MATDIM,MITER,ML,MU,N,
     8   NDE,NQ,T,USERS,Y,YH,YWT,EVALFA,SAVE1,SAVE2,A,D,JSTATE)
C***BEGIN PROLOGUE  SDCOR
C***REFER TO  SDRIV3
C  Subroutine SDCOR is called to compute corrections to the Y array.
C  In the case of functional iteration, update Y directly from the
C  result of the last call to F.
C  In the case of the chord method, compute the corrector error and
C  solve the linear system with that as right hand side and DFDY as
C  coefficient matrix, using the LU decomposition if MITER is 1, 2, 4,
C  or 5.
C***ROUTINES CALLED  SGESL,SGBSL,SNRM2
C***DATE WRITTEN   790601   (YYMMDD)
C***REVISION DATE  870401   (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C           SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDCOR
      REAL A(MATDIM,*), D, DFDY(MATDIM,*), EL(13,12), H,
     8     SAVE1(*), SAVE2(*), SNRM2, T, Y(*), YH(N,*), YWT(*)
      INTEGER IPVT(*)
      LOGICAL EVALFA
C***FIRST EXECUTABLE STATEMENT  SDCOR
      IF (MITER .EQ. 0) THEN
        DO 100 I = 1,N
 100      SAVE1(I) = (H*SAVE2(I) - YH(I,2) - SAVE1(I))/YWT(I)
        D = SNRM2(N, SAVE1, 1)/SQRT(REAL(N))
        DO 105 I = 1,N
 105      SAVE1(I) = H*SAVE2(I) - YH(I,2)
      ELSE IF (MITER .EQ. 1 .OR. MITER .EQ. 2) THEN
        IF (IMPL .EQ. 0) THEN
          DO 130 I = 1,N
 130        SAVE2(I) = H*SAVE2(I) - YH(I,2) - SAVE1(I)
        ELSE IF (IMPL .EQ. 1) THEN
          IF (EVALFA) THEN
            CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
            IF (N .EQ. 0) THEN
              JSTATE = 9
              RETURN
            END IF
          ELSE
            EVALFA = .TRUE.
          END IF
          DO 150 I = 1,N
 150        SAVE2(I) = H*SAVE2(I)
          DO 160 J = 1,N
            DO 160 I = 1,N
 160          SAVE2(I) = SAVE2(I) - A(I,J)*(YH(J,2) + SAVE1(J))
        ELSE IF (IMPL .EQ. 2) THEN
          IF (EVALFA) THEN
            CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
            IF (N .EQ. 0) THEN
              JSTATE = 9
              RETURN
            END IF
          ELSE
            EVALFA = .TRUE.
          END IF
          DO 180 I = 1,N
 180        SAVE2(I) = H*SAVE2(I) - A(I,1)*(YH(I,2) + SAVE1(I))
        END IF
        CALL SGESL (DFDY, MATDIM, N, IPVT, SAVE2, 0)
        DO 200 I = 1,N
          SAVE1(I) = SAVE1(I) + SAVE2(I)
 200      SAVE2(I) = SAVE2(I)/YWT(I)
        D = SNRM2(N, SAVE2, 1)/SQRT(REAL(N))
      ELSE IF (MITER .EQ. 4 .OR. MITER .EQ. 5) THEN
        IF (IMPL .EQ. 0) THEN
          DO 230 I = 1,N
 230        SAVE2(I) = H*SAVE2(I) - YH(I,2) - SAVE1(I)
        ELSE IF (IMPL .EQ. 1) THEN
          IF (EVALFA) THEN
          CALL FA (N, T, Y, A(ML+1,1), MATDIM, ML, MU, NDE)
          IF (N .EQ. 0) THEN
            JSTATE = 9
            RETURN
          END IF
        ELSE
          EVALFA = .TRUE.
        END IF
        DO 250 I = 1,N
 250      SAVE2(I) = H*SAVE2(I)
        MW = ML + 1 + MU
        DO 260 J = 1,N
          I1 = MAX(ML+1, MW+1-J)
          I2 = MIN(MW+N-J, MW+ML)
          DO 260 I = I1,I2
            I3 = I + J - MW
 260        SAVE2(I3) = SAVE2(I3) - A(I,J)*(YH(J,2) + SAVE1(J))
      ELSE IF (IMPL .EQ. 2) THEN
        IF (EVALFA) THEN
          CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
          IF (N .EQ. 0) THEN
            JSTATE = 9
            RETURN
          END IF
        ELSE
          EVALFA = .TRUE.
        END IF
        DO 280 I = 1,N
 280      SAVE2(I) = H*SAVE2(I) - A(I,1)*(YH(I,2) + SAVE1(I))
      END IF
      CALL SGBSL (DFDY, MATDIM, N, ML, MU, IPVT, SAVE2, 0)
      DO 300 I = 1,N
        SAVE1(I) = SAVE1(I) + SAVE2(I)
 300    SAVE2(I) = SAVE2(I)/YWT(I)
      D = SNRM2(N, SAVE2, 1)/SQRT(REAL(N))
    ELSE IF (MITER .EQ. 3) THEN
      IFLAG = 2
      CALL USERS (Y, YH(1,2), YWT, SAVE1, SAVE2, T, H, EL(1,NQ), IMPL,
   8              N, NDE, IFLAG)
      IF (N .EQ. 0) THEN
        JSTATE = 10
        RETURN
      END IF
      DO 320 I = 1,N
        SAVE1(I) = SAVE1(I) + SAVE2(I)
 320    SAVE2(I) = SAVE2(I)/YWT(I)
      D = SNRM2(N, SAVE2, 1)/SQRT(REAL(N))
    END IF
    END
    SUBROUTINE SDCST (MAXORD,MINT,ISWFLG,EL,TQ)
C***BEGIN PROLOGUE  SDCST
C***REFER TO  SDRIV3
C  SDCST is called by SDNTL and sets coefficients used by the core
C  integrator SDSTP.  The array EL determines the basic method.
C  The array TQ is involved in adjusting the step size in relation
C  to truncation error.  EL and TQ depend upon MINT, and are calculated
```

```
C  for orders 1 to MAXORD(.LE. 12).  For each order NQ, the coefficients
C  EL are calculated from the generating polynomial:
C    L(T) = EL(1,NQ) + EL(2,NQ)*T + ... + EL(NQ+1,NQ)*T**NQ.
C  For the implicit Adams methods, L(T) is given by
C    dL/dT = (1+T)*(2+T)* ... *(NQ-1+T)/K,   L(-1) = 0,
C    where      K = factorial(NQ-1).
C  For the Gear methods,
C    L(T) = (1+T)*(2+T)* ... *(NQ+T)/K,
C    where      K = factorial(NQ)*(1 + 1/2 + ... + 1/NQ).
C  For each order NQ, there are three components of TQ.
C***ROUTINES CALLED  (NONE)
C***DATE WRITTEN   790601   (YYMMDD)
C***REVISION DATE  870216   (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C           SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDCST
      REAL EL(13,12), FACTRL(12), GAMMA(14), SUM, TQ(3,12)
C***FIRST EXECUTABLE STATEMENT  SDCST
      FACTRL(1) = 1.E0
      DO 10 I = 2,MAXORD
 10     FACTRL(I) = REAL(I)*FACTRL(I-1)
C                                            COMPUTE ADAMS COEFFICIENTS
      IF (MINT .EQ. 1) THEN
        GAMMA(1) = 1.E0
        DO 40 I = 1,MAXORD+1
          SUM = 0.E0
          DO 30 J = 1,I
 30         SUM = SUM - GAMMA(J)/REAL(I-J+2)
 40       GAMMA(I+1) = SUM
        EL(1,1) = 1.E0
        EL(2,1) = 1.E0
        EL(2,2) = 1.E0
        EL(3,2) = 1.E0
        DO 60 J = 3,MAXORD
          EL(2,J) = FACTRL(J-1)
          DO 50 I = 3,J
 50         EL(I,J) = REAL(J-1)*EL(I,J-1) + EL(I-1,J-1)
 60       EL(J+1,J) = 1.E0
        DO 80 J = 2,MAXORD
          EL(1,J) = EL(1,J-1) + GAMMA(J)
          EL(2,J) = 1.E0
          DO 80 I = 3,J+1
 80         EL(I,J) = EL(I,J)/(REAL(I-1)*FACTRL(J-1))
        DO 100 J = 1,MAXORD
          TQ(1,J) = -1.E0/(FACTRL(J)*GAMMA(J))
          TQ(2,J) = -1.E0/GAMMA(J+1)
 100      TQ(3,J) = -1.E0/GAMMA(J+2)
C                                             COMPUTE GEAR COEFFICIENTS
      ELSE IF (MINT .EQ. 2) THEN
        EL(1,1) = 1.E0
        EL(2,1) = 1.E0
        DO 130 J = 2,MAXORD
          EL(1,J) = FACTRL(J)
          DO 120 I = 2,J
 120        EL(I,J) = REAL(J)*EL(I,J-1) + EL(I-1,J-1)
 130        EL(J+1,J) = 1.E0
          SUM = 1.E0
          DO 150 J = 2,MAXORD
            SUM = SUM + 1.E0/REAL(J)
            DO 150 I = 1,J+1
 150          EL(I,J) = EL(I,J)/(FACTRL(J)*SUM)
          DO 170 J = 1,MAXORD
            IF (J .GT. 1) TQ(1,J) = 1.E0/FACTRL(J-1)
            TQ(2,J) = REAL(J+1)/EL(1,J)
 170        TQ(3,J) = REAL(J+2)/EL(1,J)
        END IF
C                             Compute constants used in the stiffness test.
C                             These are the ratio of TQ(2,NQ) for the Gear
C                             methods to those for the Adams methods.
        IF (ISWFLG .EQ. 3) THEN
          MXRD = MIN(MAXORD, 5)
          IF (MINT .EQ. 2) THEN
            GAMMA(1) = 1.E0
            DO 190 I = 1,MXRD
              SUM = 0.E0
              DO 180 J = 1,I
 180            SUM = SUM - GAMMA(J)/REAL(I-J+2)
 190          GAMMA(I+1) = SUM
          END IF
          SUM = 1.E0
          DO 200 I = 2,MXRD
            SUM = SUM + 1.E0/REAL(I)
 200        EL(1+I,1) = -REAL(I+1)*SUM*GAMMA(I+1)
        END IF
        END
        SUBROUTINE SDNTL (EPS,F,FA,HMAX,HOLD,IMPL,JTASK,MATDIM,MAXORD,
     8   MINT,MITER,ML,MU,N,NDE,SAVE1,T,UROUND,USERS,Y,YWT,H,MNTOLD,
     8   MTROLD,NFE,RC,YH,A,CONVRG,EL,FAC,IER,IPVT,NQ,NWAIT,RH,RMAX,
     8   SAVE2,TQ,TREND,ISWFLG,JSTATE)
C***BEGIN PROLOGUE  SDNTL
C***REFER TO  SDRIV3
C  Subroutine SDNTL is called to set parameters on the first call
C  to SDSTP, on an internal restart, or when the user has altered
C  MINT, MITER, and/or H.
C  On the first call, the order is set to 1 and the initial derivatives
C  are calculated.  RMAX is the maximum ratio by which H can be
C  increased in one step.  It is initially RMINIT to compensate
C  for the small initial H, but then is normally equal to RMNORM.
C  If a failure occurs (in corrector convergence or error test), RMAX
C  is set at RMFAIL for the next increase.
C  If the caller has changed MINT, or if JTASK = 0, SDCST is called
C  to set the coefficients of the method.  If the caller has changed H,
C  YH must be rescaled.  If H or MINT has been changed, NWAIT is
C  reset to NQ + 2 to prevent further increases in H for that many
C  steps.  Also, RC is reset.  RC is the ratio of new to old values of
C  the coefficient L(0)*H.  If the caller has changed MITER, RC is
C  set to 0 to force the partials to be updated, if partials are used.
C***ROUTINES CALLED  SDCST,SDSCL,SGEFA,SGESL,SGBFA,SGBSL,SNRM2
C***DATE WRITTEN   790601   (YYMMDD)
C***REVISION DATE  870810   (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
```

```
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C           SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDNTL
      REAL A(MATDIM,*), EL(13,12), EPS, FAC(*), H, HMAX,
     8     HOLD, OLDLO, RC, RH, RMAX, RMINIT, SAVE1(*), SAVE2(*), SMAX,
     8     SMIN, SNRM2, SUM, SUM0, T, TQ(3,12), TREND, UROUND, Y(*),
     8     YH(N,*), YWT(*)
      INTEGER IPVT(*)
      LOGICAL CONVRG, IER
      PARAMETER(RMINIT = 10000.E0)
C***FIRST EXECUTABLE STATEMENT  SDNTL
      IER = .FALSE.
      IF (JTASK .GE. 0) THEN
        IF (JTASK .EQ. 0) THEN
          CALL SDCST (MAXORD, MINT, ISWFLG,  EL, TQ)
          RMAX = RMINIT
        END IF
        RC = 0.E0
        CONVRG = .FALSE.
        TREND = 1.E0
        NQ = 1
        NWAIT = 3
        CALL F (N, T, Y, SAVE2)
        IF (N .EQ. 0) THEN
          JSTATE = 6
          RETURN
        END IF
        NFE = NFE + 1
        IF (IMPL .NE. 0) THEN
          IF (MITER .EQ. 3) THEN
            IFLAG = 0
            CALL USERS (Y, YH, YWT, SAVE1, SAVE2, T, H, EL, IMPL, N,
     8                  NDE, IFLAG)
            IF (N .EQ. 0) THEN
              JSTATE = 10
              RETURN
            END IF
          ELSE IF (IMPL .EQ. 1) THEN
            IF (MITER .EQ. 1 .OR. MITER .EQ. 2) THEN
              CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
              IF (N .EQ. 0) THEN
                JSTATE = 9
                RETURN
              END IF
              CALL SGEFA (A, MATDIM, N, IPVT, INFO)
              IF (INFO .NE. 0) THEN
                IER = .TRUE.
                RETURN
              END IF
              CALL SGESL (A, MATDIM, N, IPVT, SAVE2, 0)
            ELSE IF (MITER .EQ. 4 .OR. MITER .EQ. 5) THEN
              CALL FA (N, T, Y, A(ML+1,1), MATDIM, ML, MU, NDE)
              IF (N .EQ. 0) THEN
                JSTATE = 9
                RETURN
              END IF
              CALL SGBFA (A, MATDIM, N, ML, MU, IPVT, INFO)
              IF (INFO .NE. 0) THEN
                IER = .TRUE.
                RETURN
              END IF
              CALL SGBSL (A, MATDIM, N, ML, MU, IPVT, SAVE2, 0)
            END IF
          ELSE IF (IMPL .EQ. 2) THEN
            CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
            IF (N .EQ. 0) THEN
              JSTATE = 9
              RETURN
            END IF
            DO 150 I = 1,NDE
              IF (A(I,1) .EQ. 0.E0) THEN
                IER = .TRUE.
                RETURN
              ELSE
                SAVE2(I) = SAVE2(I)/A(I,1)
              END IF
 150          CONTINUE
            DO 155 I = NDE+1,N
 155          A(I,1) = 0.E0
          END IF
        END IF
        DO 170 I = 1,NDE
 170      SAVE1(I) = SAVE2(I)/YWT(I)
        SUM = SNRM2(NDE, SAVE1, 1)
        SUM0 = 1.E0/MAX(1.E0, ABS(T))
        SMAX = MAX(SUM0, SUM)
        SMIN = MIN(SUM0, SUM)
        SUM = SMAX*SQRT(1.E0 + (SMIN/SMAX)**2)/SQRT(REAL(NDE))
        H = SIGN(MIN(2.E0*EPS/SUM, ABS(H)), H)
        DO 180 I = 1,N
 180      YH(I,2) = H*SAVE2(I)
        IF (MITER .EQ. 2 .OR. MITER .EQ. 5 .OR. ISWFLG .EQ. 3) THEN
          DO 20 I = 1,N
 20         FAC(I) = SQRT(UROUND)
        END IF
      ELSE
        IF (MITER .NE. MTROLD) THEN
          MTROLD = MITER
          RC = 0.E0
          CONVRG = .FALSE.
        END IF
        IF (MINT .NE. MNTOLD) THEN
          MNTOLD = MINT
          OLDL0 = EL(1,NQ)
          CALL SDCST (MAXORD, MINT, ISWFLG,  EL, TQ)
          RC = RC*EL(1,NQ)/OLDL0
          NWAIT = NQ + 2
        END IF
        IF (H .NE. HOLD) THEN
          NWAIT = NQ + 2
          RH = H/HOLD
          CALL SDSCL (HMAX, N, NQ, RMAX,  HOLD, RC, RH, YH)
        END IF
      END IF
      END
      SUBROUTINE SDNTP (H,K,N,NQ,T,TOUT,YH,Y)
C***BEGIN PROLOGUE  SDNTP
C***REFER TO  SDRIV3
C   Subroutine SDNTP interpolates the K-th derivative of Y at TOUT,
C   using the data in the YH array.  If K has a value greater than NQ,
C   the NQ-th derivative is calculated.
C***ROUTINES CALLED  (NONE)
C***DATE WRITTEN   790601   (YYMMDD)
C***REVISION DATE  870216   (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C           SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDNTP
      REAL FACTOR, H, R, T, TOUT, Y(*), YH(N,*)
C***FIRST EXECUTABLE STATEMENT  SDNTP
      IF (K .EQ. 0) THEN
        DO 10 I = 1,N
 10       Y(I) = YH(I,NQ+1)
        R = ((TOUT - T)/H)
        DO 20 JJ = 1,NQ
          J = NQ + 1 - JJ
          DO 20 I = 1,N
 20         Y(I) = YH(I,J) + R*Y(I)
      ELSE
        KUSED = MIN(K, NQ)
        FACTOR = 1.E0
        DO 40 KK = 1,KUSED
 40       FACTOR = FACTOR*REAL(NQ+1-KK)
        DO 50 I = 1,N
 50       Y(I) = FACTOR*YH(I,NQ+1)
        DO 80 JJ = KUSED+1,NQ
          J = K + 1 + NQ - JJ
          FACTOR = 1.E0
          DO 60 KK = 1,KUSED
 60         FACTOR = FACTOR*REAL(J-KK)
          DO 70 I = 1,N
 70         Y(I) = FACTOR*YH(I,J) + R*Y(I)
 80       CONTINUE
        DO 100 I = 1,N
 100      Y(I) = Y(I)*H**(-KUSED)
      END IF
      END
      REAL FUNCTION SDOT(N,SX,INCX,SY,INCY)
C
C     RETURNS THE DOT PRODUCT OF SINGLE PRECISION SX AND SY.
C     SDOT = SUM FOR I = 0 TO N-1 OF  SX(LX+I*INCX) * SY(LY+I*INCY),
C     WHERE LX = 1 IF INCX .GE. 0, ELSE LX = (-INCX)*N, AND LY IS
C     DEFINED IN A SIMILAR WAY USING INCY.
C
      REAL SX(1),SY(1)
      SDOT = 0.0E0
      IF(N.LE.0)RETURN
      IF(INCX.EQ.INCY) IF(INCX-1)5,20,60
```

```
    5 CONTINUE
C
C        CODE FOR UNEQUAL INCREMENTS OR NONPOSITIVE INCREMENTS.
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        SDOT = SDOT + SX(IX)*SY(IY)
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      RETURN
C
C        CODE FOR BOTH INCREMENTS EQUAL TO 1
C
C
C        CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 5.
C
   20 M = MOD(N,5)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        SDOT = SDOT + SX(I)*SY(I)
   30 CONTINUE
      IF( N .LT. 5 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,5
        SDOT = SDOT + SX(I)*SY(I) + SX(I + 1)*SY(I + 1) +
     1   SX(I + 2)*SY(I + 2) + SX(I + 3)*SY(I + 3) + SX(I + 4)*SY(I + 4)
   50 CONTINUE
      RETURN
C
C        CODE FOR POSITIVE EQUAL INCREMENTS .NE.1.
C
   60 CONTINUE
      NS=N*INCX
      DO 70 I=1,NS,INCX
        SDOT = SDOT + SX(I)*SY(I)
   70   CONTINUE
      RETURN
      END
      SUBROUTINE SDPSC (KSGN,N,NQ,YH)
C***BEGIN PROLOGUE  SDPSC
C***REFER TO  SDRIV3
C     This subroutine computes the predicted YH values by effectively
C     multiplying the YH array by the Pascal triangle matrix when KSGN
C     is +1, and performs the inverse function when KSGN is -1.
C***ROUTINES CALLED  (NONE)
C***DATE WRITTEN   790601   (YYMMDD)
C***REVISION DATE  841119   (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C           SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDPSC
      REAL YH(N,*)
```

```
C***FIRST EXECUTABLE STATEMENT  SDPSC
      IF (KSGN .GT. 0) THEN
        DO 10 J1 = 1,NQ
          DO 10 J2 = J1,NQ
            J = NQ - J2 + J1
            DO 10 I = 1,N
 10           YH(I,J) = YH(I,J) + YH(I,J+1)
      ELSE
        DO 30 J1 = 1,NQ
          DO 30 J2 = J1,NQ
            J = NQ - J2 + J1
            DO 30 I = 1,N
 30               YH(I,J) = YH(I,J) - YH(I,J+1)
      END IF
      RETURN
      END
      SUBROUTINE SGBFA(ABD,LDA,N,ML,MU,IPVT,INFO)
C***BEGIN PROLOGUE  SGBFA
C***DATE WRITTEN   780814   (YYMMDD)
C***REVISION DATE  820801   (YYMMDD)
C***CATEGORY NO.  D2A2
C***KEYWORDS  BANDED,FACTOR,LINEAR ALGEBRA,LINPACK,MATRIX
C***AUTHOR  MOLER, C. B., (U. OF NEW MEXICO)
C***PURPOSE  Factors a real BAND matrix by elimination.
C***DESCRIPTION
C
C     SGBFA factors a real band matrix by elimination.
C
C     SGBFA is usually called by SBGCO, but it can be called
C     directly with a saving in time if  RCOND  is not needed.
C
C     LINPACK.  This version dated 08/14/78 .
C     Cleve Moler, University of New Mexico, Argonne National Lab.
C
C     Subroutines and Functions
C
C     BLAS SAXPY,SSCAL,ISAMAX
C     Fortran MAX0,MIN0
C***REFERENCES  DONGARRA J.J., BUNCH J.R., MOLER C.B., STEWART G.W.,
C                 *LINPACK USERS  GUIDE*, SIAM, 1979.
C***ROUTINES CALLED  ISAMAX,SAXPY,SSCAL
C***END PROLOGUE  SGBFA
      INTEGER LDA,N,ML,MU,IPVT(1),INFO
      REAL ABD(LDA,1)
C
      REAL T
      INTEGER I,ISAMAX,I0,J,JU,JZ,J0,J1,K,KP1,L,LM,M,MM,NM1
C
C***FIRST EXECUTABLE STATEMENT  SGBFA
      M = ML + MU + 1
      INFO = 0
C
C     ZERO INITIAL FILL-IN COLUMNS
C
      J0 = MU + 2
      J1 = MIN0(N,M) - 1
      IF (J1 .LT. J0) GO TO 30
      DO 20 JZ = J0, J1
         I0 = M + 1 - JZ
         DO 10 I = I0, ML
            ABD(I,JZ) = 0.0E0
   10    CONTINUE
   20 CONTINUE
   30 CONTINUE
      JZ = J1
      JU = 0
C
C     GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
C
      NM1 = N - 1
      IF (NM1 .LT. 1) GO TO 130
      DO 120 K = 1, NM1
         KP1 = K + 1
C
C        ZERO NEXT FILL-IN COLUMN
C
         JZ = JZ + 1
         IF (JZ .GT. N) GO TO 50
         IF (ML .LT. 1) GO TO 50
            DO 40 I = 1, ML
               ABD(I,JZ) = 0.0E0
   40       CONTINUE
   50    CONTINUE
C
C        FIND L = PIVOT INDEX
C
         LM = MIN0(ML,N-K)
         L = ISAMAX(LM+1,ABD(M,K),1) + M - 1
         IPVT(K) = L + K - M
C
C        ZERO PIVOT IMPLIES THIS COLUMN ALREADY TRIANGULARIZED
C
         IF (ABD(L,K) .EQ. 0.0E0) GO TO 100
C
C           INTERCHANGE IF NECESSARY
C
            IF (L .EQ. M) GO TO 60
               T = ABD(L,K)
               ABD(L,K) = ABD(M,K)
               ABD(M,K) = T
   60       CONTINUE
C
C           COMPUTE MULTIPLIERS
C
            T = -1.0E0/ABD(M,K)
            CALL SSCAL(LM,T,ABD(M+1,K),1)
C
C           ROW ELIMINATION WITH COLUMN INDEXING
C
            JU = MIN0(MAX0(JU,MU+IPVT(K)),N)
            MM = M
            IF (JU .LT. KP1) GO TO 90
            DO 80 J = KP1, JU
               L = L - 1
               MM = MM - 1
               T = ABD(L,J)
               IF (L .EQ. MM) GO TO 70
                  ABD(L,J) = ABD(MM,J)
                  ABD(MM,J) = T
   70          CONTINUE
               CALL SAXPY(LM,T,ABD(M+1,K),1,ABD(MM+1,J),1)
   80       CONTINUE
   90       CONTINUE
         GO TO 110
  100    CONTINUE
            INFO = K
  110    CONTINUE
  120 CONTINUE
  130 CONTINUE
      IPVT(N) = N
      IF (ABD(M,N) .EQ. 0.0E0) INFO = N
      RETURN
      END
      SUBROUTINE SDPST (EL,F,FA,H,IMPL,JACOBN,MATDIM,MITER,ML,MU,N,NDE,
     8   NQ,SAVE2,T,USERS,Y,YH,YWT,UROUND,NFE,NJE,A,DFDY,FAC,IER,IPVT,
     8   SAVE1,ISWFLG,BND,JSTATE)
C***BEGIN PROLOGUE  SDPST
C***REFER TO  SDRIV3
C  Subroutine SDPST is called to reevaluate the partials.
C  If MITER is 1, 2, 4, or 5, the matrix
C  P = I - L(0)*H*Jacobian is stored in DFDY and subjected to LU
C  decomposition, with the results also stored in DFDY.
C***ROUTINES CALLED  SGEFA,SGBFA,SNRM2
C***DATE WRITTEN   790601   (YYMMDD)
C***REVISION DATE  870401   (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C           SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDPST
      REAL A(MATDIM,*), BL, BND, BP, BR, BU, DFDY(MATDIM,*),
     8     DFDYMX, DIFF, DY, EL(13,12), FAC(*), FACMAX, FACMIN, FACTOR,
     8     H, SAVE1(*), SAVE2(*), SCALE, SNRM2, T, UROUND, Y(*),
     8     YH(N,*), YJ, YS, YWT(*)
      INTEGER IPVT(*)
      LOGICAL IER
      PARAMETER(FACMAX = .5E0)
C***FIRST EXECUTABLE STATEMENT  SDPST
      NJE = NJE + 1
      IER = .FALSE.
      IF (MITER .EQ. 1 .OR. MITER .EQ. 2) THEN
        IF (MITER .EQ. 1) THEN
          CALL JACOBN (N, T, Y, DFDY, MATDIM, ML, MU)
          IF (N .EQ. 0) THEN
            JSTATE = 8
            RETURN
          END IF
          IF (ISWFLG .EQ. 3) BND = SNRM2(N*N, DFDY, 1)
          FACTOR = -EL(1,NQ)*H
          DO 110 J = 1,N
            DO 110 I = 1,N
110           DFDY(I,J) = FACTOR*DFDY(I,J)
        ELSE IF (MITER .EQ. 2) THEN
          BR = UROUND**(.875E0)
          BL = UROUND**(.75E0)
          BU = UROUND**(.25E0)
          BP = UROUND**(-.15E0)
          FACMIN = UROUND**(.78E0)
          DO 170 J = 1,N
            YS = MAX(ABS(YWT(J)), ABS(Y(J)))
120         DY = FAC(J)*YS
            IF (DY .EQ. 0.E0) THEN
              IF (FAC(J) .LT. FACMAX) THEN
                FAC(J) = MIN(100.E0*FAC(J), FACMAX)
                GO TO 120
              ELSE
                DY = YS
              END IF
            END IF
            IF (NQ .EQ. 1) THEN
              DY = SIGN(DY, SAVE2(J))
            ELSE
              DY = SIGN(DY, YH(J,3))
            END IF
            DY = (Y(J) + DY) - Y(J)
            YJ = Y(J)
            Y(J) = Y(J) + DY
            CALL F (N, T, Y, SAVE1)
            IF (N .EQ. 0) THEN
              JSTATE = 6
              RETURN
            END IF
            Y(J) = YJ
            FACTOR = -EL(1,NQ)*H/DY
            DO 140 I = 1,N
140           DFDY(I,J) = (SAVE1(I) - SAVE2(I))*FACTOR
C                                                                Step 1
            DIFF = ABS(SAVE2(1) - SAVE1(1))
            IMAX = 1
            DO 150 I = 2,N
              IF (ABS(SAVE2(I) - SAVE1(I)) .GT. DIFF) THEN
                IMAX = I
                DIFF = ABS(SAVE2(I) - SAVE1(I))
              END IF
150           CONTINUE
C                                                                Step 2
            IF (MIN(ABS(SAVE2(IMAX)), ABS(SAVE1(IMAX))) .GT. 0.E0) THEN
              SCALE = MAX(ABS(SAVE2(IMAX)), ABS(SAVE1(IMAX)))
C                                                                Step 3
              IF (DIFF .GT. BU*SCALE) THEN
                FAC(J) = MAX(FACMIN, FAC(J)*.1E0)
              ELSE IF (BR*SCALE .LE. DIFF .AND. DIFF .LE. BL*SCALE) THEN
                FAC(J) = MIN(FAC(J)*10.E0, FACMAX)
C                                                                Step 4
              ELSE IF (DIFF .LT. BR*SCALE) THEN
```

```
                FAC(J) = MIN(BP*FAC(J), FACMAX)
              END IF
            END IF
 170        CONTINUE
          IF (ISWFLG .EQ. 3) BND = SNRM2(N*N, DFDY, 1)/(-EL(1,NQ)*H)
          NFE = NFE + N
        END IF
        IF (IMPL .EQ. 0) THEN
          DO 190 I = 1,N
 190        DFDY(I,I) = DFDY(I,I) + 1.E0
        ELSE IF (IMPL .EQ. 1) THEN
          CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
          IF (N .EQ. 0) THEN
            JSTATE = 9
            RETURN
          END IF
          DO 210 J = 1,N
            DO 210 I = 1,N
 210          DFDY(I,J) = DFDY(I,J) + A(I,J)
        ELSE IF (IMPL .EQ. 2) THEN
          CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
          IF (N .EQ. 0) THEN
            JSTATE = 9
            RETURN
          END IF
          DO 230 I = 1,NDE
 230        DFDY(I,I) = DFDY(I,I) + A(I,1)
        END IF
        CALL SGEFA (DFDY, MATDIM, N, IPVT, INFO)
        IF (INFO .NE. 0) IER = .TRUE.
      ELSE IF (MITER .EQ. 4 .OR. MITER .EQ. 5) THEN
        IF (MITER .EQ. 4) THEN
          CALL JACOBN (N, T, Y, DFDY(ML+1,1), MATDIM, ML, MU)
          IF (N .EQ. 0) THEN
            JSTATE = 8
            RETURN
          END IF
          FACTOR = -EL(1,NQ)*H
          MW = ML + MU + 1
          DO 260 J = 1,N
            I1 = MAX(ML+1, MW+1-J)
            I2 = MIN(MW+N-J, MW+ML)
            DO 260 I = I1,I2
 260          DFDY(I,J) = FACTOR*DFDY(I,J)
        ELSE IF (MITER .EQ. 5) THEN
          BR = UROUND**(.875E0)
          BL = UROUND**(.75E0)
          BU = UROUND**(.25E0)
          BP = UROUND**(-.15E0)
          FACMIN = UROUND**(.78E0)
          MW = ML + MU + 1
          J2 = MIN(MW, N)
          DO 340 J = 1,J2
            DO 290 K = J,N,MW
              YS = MAX(ABS(YWT(K)), ABS(Y(K)))
 280          DY = FAC(K)*YS
```

```
              IF (DY .EQ. 0.E0) THEN
                IF (FAC(K) .LT. FACMAX) THEN
                  FAC(K) = MIN(100.E0*FAC(K), FACMAX)
                  GO TO 280
                ELSE
                  DY = YS
                END IF
              END IF
              IF (NQ .EQ. 1) THEN
                DY = SIGN(DY, SAVE2(K))
              ELSE
                DY = SIGN(DY, YH(K,3))
              END IF
              DY = (Y(K) + DY) - Y(K)
              DFDY(MW,K) = Y(K)
 290          Y(K) = Y(K) + DY
            CALL F (N, T, Y, SAVE1)
            IF (N .EQ. 0) THEN
              JSTATE = 6
              RETURN
            END IF
            DO 330 K = J,N,MW
              Y(K) = DFDY(MW,K)
              YS = MAX(ABS(YWT(K)), ABS(Y(K)))
              DY = FAC(K)*YS
              IF (DY .EQ. 0.E0) DY = YS
              IF (NQ .EQ. 1) THEN
                DY = SIGN(DY, SAVE2(K))
              ELSE
                DY = SIGN(DY, YH(K,3))
              END IF
              DY = (Y(K) + DY) - Y(K)
              FACTOR = -EL(1,NQ)*H/DY
              I1 = MAX(ML+1, MW+1-K)
              I2 = MIN(MW+N-K, MW+ML)
              DO 300 I = I1,I2
                I3 = K + I - MW
 300            DFDY(I,K) = FACTOR*(SAVE1(I3) - SAVE2(I3))
C                                                                  Step 1
              IMAX = MAX(1, K - MU)
              DIFF = ABS(SAVE2(IMAX) - SAVE1(IMAX))
              I1 = IMAX
              I2 = MIN(K + ML, N)
              DO 310 I = I1+1,I2
                IF (ABS(SAVE2(I) - SAVE1(I)) .GT. DIFF) THEN
                  IMAX = I
                  DIFF = ABS(SAVE2(I) - SAVE1(I))
                END IF
 310            CONTINUE
C                                                                  Step 2
              IF (MIN(ABS(SAVE2(IMAX)), ABS(SAVE1(IMAX))) .GT.0.E0) THEN
                SCALE = MAX(ABS(SAVE2(IMAX)), ABS(SAVE1(IMAX)))
C                                                                  Step 3
                IF (DIFF .GT. BU*SCALE) THEN
                  FAC(K) = MAX(FACMIN, FAC(K)*.1E0)
                ELSE IF (BR*SCALE .LE.DIFF .AND. DIFF .LE.BL*SCALE) THEN
```

```
                  FAC(K) = MIN(FAC(K)*10.E0, FACMAX)
C                                                                     Step 4
                ELSE IF (DIFF .LT. BR*SCALE) THEN
                  FAC(K) = MIN(BP*FAC(K), FACMAX)
                END IF
              END IF
 330          CONTINUE
 340        CONTINUE
          NFE = NFE + J2
        END IF
        IF (ISWFLG .EQ. 3) THEN
          DFDYMX = 0.E0
          DO 345 J = 1,N
            I1 = MAX(ML+1, MW+1-J)
            I2 = MIN(MW+N-J, MW+ML)
            DO 345 I = I1,I2
 345          DFDYMX = MAX(DFDYMX, ABS(DFDY(I,J)))
          BND = 0.E0
          IF (DFDYMX .NE. 0.E0) THEN
            DO 350 J = 1,N
              I1 = MAX(ML+1, MW+1-J)
              I2 = MIN(MW+N-J, MW+ML)
              DO 350 I = I1,I2
 350            BND = BND + (DFDY(I,J)/DFDYMX)**2
            BND = DFDYMX*SQRT(BND)/(-EL(1,NQ)*H)
          END IF
        END IF
        IF (IMPL .EQ. 0) THEN
          DO 360 J = 1,N
 360        DFDY(MW,J) = DFDY(MW,J) + 1.E0
        ELSE IF (IMPL .EQ. 1) THEN
          CALL FA (N, T, Y, A(ML+1,1), MATDIM, ML, MU, NDE)
          IF (N .EQ. 0) THEN
            JSTATE = 9
            RETURN
          END IF
          DO 380 J = 1,N
            I1 = MAX(ML+1, MW+1-J)
            I2 = MIN(MW+N-J, MW+ML)
            DO 380 I = I1,I2
 380          DFDY(I,J) = DFDY(I,J) + A(I,J)
        ELSE IF (IMPL .EQ. 2) THEN
          CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
          IF (N .EQ. 0) THEN
            JSTATE = 9
            RETURN
          END IF
          DO 400 J = 1,NDE
 400        DFDY(MW,J) =  DFDY(MW,J) + A(J,1)
        END IF
        CALL SGBFA (DFDY, MATDIM, N, ML, MU, IPVT, INFO)
        IF (INFO .NE. 0) IER = .TRUE.
      ELSE IF (MITER .EQ. 3) THEN
        IFLAG = 1
        CALL USERS (Y, YH(1,2), YWT, SAVE1, SAVE2, T, H, EL(1,NQ), IMPL,
     8              N, NDE, IFLAG)
```

```
        IF (N .EQ. 0) THEN
          JSTATE = 10
          RETURN
        END IF
      END IF
      END
      SUBROUTINE SDRIV1 (N,T,Y,TOUT,MSTATE,EPS,WORK,LENW)
C***BEGIN PROLOGUE  SDRIV1
C***DATE WRITTEN   790601   (YYMMDD)
C***REVISION DATE  870401   (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***KEYWORDS  ODE,STIFF,ORDINARY DIFFERENTIAL EQUATIONS,
C             INITIAL VALUE PROBLEMS,GEAR'S METHOD,
C             SINGLE PRECISION
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C           SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***PURPOSE  The function of SDRIV1 is to solve N (200 or fewer)
C            ordinary differential equations of the form
C            dY(I)/dT = F(Y(I),T), given the initial conditions
C            Y(I) = YI.  SDRIV1 uses single precision arithmetic.
C***DESCRIPTION
C
C   Version 87.1
C
C  I.  CHOOSING THE CORRECT ROUTINE  ...............................
C
C     SDRIV
C     DDRIV
C     CDRIV
C           These are the generic names for three packages for solving
C           initial value problems for ordinary differential equations.
C           SDRIV uses single precision arithmetic.  DDRIV uses double
C           precision arithmetic.  CDRIV allows complex-valued
C           differential equations, integrated with respect to a single,
C           real, independent variable.
C
C    As an aid in selecting the proper program, the following is a
C    discussion of the important options or restrictions associated with
C    each program:
C
C      A. SDRIV1 should be tried first for those routine problems with
C         no more than 200 differential equations.  Internally this
C         routine has two important technical defaults:
C           1. Numerical approximation of the Jacobian matrix of the
C              right hand side is used.
C           2. The stiff solver option is used.
C         Most users of SDRIV1 should not have to concern themselves
C         with these details.
C
C      B. SDRIV2 should be considered for those problems for which
C         SDRIV1 is inadequate (SDRIV2 has no explicit restriction on
C         the number of differential equations.)  For example, SDRIV1
C         may have difficulty with problems having zero initial
C         conditions and zero derivatives.  In this case SDRIV2, with an
C         appropriate value of the parameter EWT, should perform more
C         efficiently.  SDRIV2 provides three important additional
```

```
C         options:
C           1. The nonstiff equation solver (as well as the stiff
C              solver) is available.
C           2. The root-finding option is available.
C           3. The program can dynamically select either the non-stiff
C              or the stiff methods.
C         Internally this routine also defaults to the numerical
C         approximation of the Jacobian matrix of the right hand side.
C
C      C. SDRIV3 is the most flexible, and hence the most complex, of
C         the programs.  Its important additional features include:
C           1. The ability to exploit band structure in the Jacobian
C              matrix.
C           2. The ability to solve some implicit differential
C              equations, i.e., those having the form:
C                   A(Y,T)*dY/dT = F(Y,T).
C           3. The option of integrating in the one step mode.
C           4. The option of allowing the user to provide a routine
C              which computes the analytic Jacobian matrix of the right
C              hand side.
C           5. The option of allowing the user to provide a routine
C              which does all the matrix algebra associated with
C              corrections to the solution components.
C
C
C***REFERENCES  GEAR, C. W., "NUMERICAL INITIAL VALUE PROBLEMS IN
C                 ORDINARY DIFFERENTIAL EQUATIONS", PRENTICE-HALL, 1971.
C***ROUTINES CALLED  SDR31,XERROR
C***END PROLOGUE  SDRIV1
      EXTERNAL F
      REAL EPS, EWT, HMAX, T, TOUT, WORK(*), Y(*)
      PARAMETER(MXN = 200, IDLIW = 21, MXLIW = IDLIW + MXN)
      INTEGER IWORK(MXLIW)
      CHARACTER MSG*103
      PARAMETER(NROOT = 0, EWT = 1.E0, IERROR = 2, MINT = 2, MITER = 2,
     8          IMPL = 0, MXORD = 5, MXSTEP = 1000)
C***FIRST EXECUTABLE STATEMENT  SDRIV1
      IF (N .GT. MXN) THEN
        WRITE(MSG, '(''SDRIV115FE Illegal input.  The number of '',
     8  ''equations,'', I8, '', is greater than the maximum allowed.'')
     8  ') N
        CALL XERROR(MSG(1:97), 97, 15, 2)
        RETURN
      END IF
      IF (MSTATE .GT. 0) THEN
        NSTATE = MSTATE
        NTASK = 1
      ELSE
        NSTATE = - MSTATE
        NTASK = 3
      END IF
      HMAX = 2.E0*ABS(TOUT - T)
      LENIW = N + IDLIW
      LENWCM = LENW - LENIW
      IF (LENWCM .LT. (N*N + 10*N + 204)) THEN
        LNWCHK = N*N + 10*N + 204 + LENIW
        WRITE(MSG, '(''SDRIV116FE Insufficient storage allocated for '',
     8  ''the work array.  The required storage is at least'', I8)')
     8  LNWCHK
        CALL XERROR(MSG(1:103), 103, 16, 2)
        RETURN
      END IF
      IF (NSTATE .NE. 1) THEN
        DO 20 I = 1,LENIW
          II = I + LENWCM
 20       IWORK(I) = INT(WORK(II))
      END IF
      CALL SDR31 (N, T, Y, F, NSTATE, TOUT, NTASK, NROOT, EPS, EWT,
     8            IERROR, MINT, MITER, IMPL, MXORD, HMAX, WORK, LENWCM,
     8            IWORK, LENIW, MXSTEP)
      DO 40 I = 1,LENIW
        II = LENWCM + I
 40     WORK(II) = REAL(IWORK(I))
      IF (NSTATE .LE. 4) THEN
        MSTATE = SIGN(NSTATE, MSTATE)
      ELSE IF (NSTATE .EQ. 6) THEN
        MSTATE = SIGN(5, MSTATE)
      END IF
      END

      SUBROUTINE SDRIV2 (N,T,Y,F,TOUT,MSTATE,NROOT,EPS,EWT,MINT,WORK,
     8   LENW,IWORK,LENIW,G)
C***BEGIN PROLOGUE  SDRIV2
C***DATE WRITTEN   790601   (YYMMDD)
C***REVISION DATE  870401   (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***KEYWORDS  ODE,STIFF,ORDINARY DIFFERENTIAL EQUATIONS,
C             INITIAL VALUE PROBLEMS,GEAR'S METHOD,
C             SINGLE PRECISION
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C           SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***PURPOSE  The function of SDRIV2 is to solve N ordinary differential
C            equations of the form dY(I)/dT = F(Y(I),T), given the
C            initial conditions Y(I) = YI.  The program has options to
C            allow the solution of both stiff and non-stiff differential
C            equations.  SDRIV2 uses single precision arithmetic.
C***DESCRIPTION
C
C  I.  ABSTRACT  .......................................................
C
C    The function of SDRIV2 is to solve N ordinary differential
C    equations of the form dY(I)/dT = F(Y(I),T), given the initial
C    conditions Y(I) = YI.  The program has options to allow the
C    solution of both stiff and non-stiff differential equations.
C    SDRIV2 is to be called once for each output point of T.
C
C***REFERENCES  GEAR, C. W., "NUMERICAL INITIAL VALUE PROBLEMS IN
C                 ORDINARY DIFFERENTIAL EQUATIONS", PRENTICE-HALL, 1971.
C***ROUTINES CALLED  SDR32, XERROR
C***END PROLOGUE  SDRIV2
      EXTERNAL F, G
      REAL EPS, EWT, EWTCOM(1), G, HMAX, T, TOUT,
     8     WORK(*), Y(*)
      INTEGER IWORK(*)
      CHARACTER MSG*81
      PARAMETER(IMPL = 0, MXSTEP = 1000)
C***FIRST EXECUTABLE STATEMENT  SDRIV2
      IF (MINT .LT. 1 .OR. MINT .GT. 3) THEN
        WRITE(MSG, '(''SDRIV21FE Illegal input.  Improper value for '',
     8  ''the integration method flag,'', I8)') MINT
        CALL XERROR(MSG(1:81), 81, 21, 2)
        RETURN
      END IF
      IF (MSTATE .GE. 0) THEN
        NSTATE = MSTATE
        NTASK = 1
      ELSE
        NSTATE = - MSTATE
        NTASK = 3
      END IF
      EWTCOM(1) = EWT
      IF (EWT .NE. 0.E0) THEN
        IERROR = 3
      ELSE
        IERROR = 2
      END IF
      IF (MINT .EQ. 1) THEN
        MITER = 0
        MXORD = 12
      ELSE IF (MINT .EQ. 2) THEN
        MITER = 2
        MXORD = 5
      ELSE IF (MINT .EQ. 3) THEN
        MITER = 2
        MXORD = 12
      END IF
      HMAX = 2.E0*ABS(TOUT - T)
      CALL SDR32 (N, T, Y, F, NSTATE, TOUT, NTASK, NROOT, EPS, EWTCOM,
     8            IERROR, MINT, MITER, IMPL, MXORD, HMAX, WORK, LENW,
     8            IWORK, LENIW, MXSTEP, G)
      IF (MSTATE .GE. 0) THEN
        MSTATE = NSTATE
      ELSE
        MSTATE = - NSTATE
      END IF
      END
      SUBROUTINE SDRIV3 (N,T,Y,F,NSTATE,TOUT,NTASK,NROOT,EPS,EWT,IERROR,
     8   MINT,MITER,IMPL,ML,MU,MXORD,HMAX,WORK,LENW,IWORK,LENIW,JACOBN,
     8   FA,NDE,MXSTEP,G,USERS)
C***BEGIN PROLOGUE  SDRIV3
C***DATE WRITTEN   790601   (YYMMDD)
C***REVISION DATE  870401   (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***KEYWORDS  ODE,STIFF,ORDINARY DIFFERENTIAL EQUATIONS,
C             INITIAL VALUE PROBLEMS,GEAR'S METHOD,
C             SINGLE PRECISION
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C           SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
```

```
C***PURPOSE  The function of SDRIV3 is to solve N ordinary differential
C            equations of the form dY(I)/dT = F(Y(I),T), given the
C            initial conditions Y(I) = YI.  The program has options to
C            allow the solution of both stiff and non-stiff differential
C            equations.  Other important options are available.  SDRIV3
C            uses single precision arithmetic.
C***DESCRIPTION
C
C  I.  ABSTRACT  .......................................................
C
C    The primary function of SDRIV3 is to solve N ordinary differential
C    equations of the form dY(I)/dT = F(Y(I),T), given the initial
C    conditions Y(I) = YI.  The program has options to allow the
C    solution of both stiff and non-stiff differential equations.  In
C    addition, SDRIV3 may be used to solve:
C      1. The initial value problem, A*dY(I)/dT = F(Y(I),T), where A is
C         a non-singular matrix depending on Y and T.
C      2. The hybrid differential/algebraic initial value problem,
C         A*dY(I)/dT = F(Y(I),T), where A is a vector (whose values may
C         depend upon Y and T) some of whose components will be zero
C         corresponding to those equations which are algebraic rather
C         than differential.
C    SDRIV3 is to be called once for each output point of T.
C
C***REFERENCES  GEAR, C. W., "NUMERICAL INITIAL VALUE PROBLEMS IN
C                 ORDINARY DIFFERENTIAL EQUATIONS", PRENTICE-HALL, 1971.
C***ROUTINES CALLED  SDSTP,SDNTP,SDZRO,SGEFA,SGESL,SGBFA,SGBSL,SNRM2,
C                    R1MACH,XERROR
C***END PROLOGUE  SDRIV3
      ENTRY SDR31 (N,T,Y,F,NSTATE,TOUT,NTASK,NROOT,EPS,EWT,IERROR,MINT,
     8   MITER,IMPL,MXORD,HMAX,WORK,LENW,IWORK,LENIW,MXSTEP)
      ENTRY SDR32 (N,T,Y,F,NSTATE,TOUT,NTASK,NROOT,EPS,EWT,IERROR,MINT,
     8   MITER,IMPL,MXORD,HMAX,WORK,LENW,IWORK,LENIW,MXSTEP,G)
      EXTERNAL F, JACOBN, FA, G, USERS
      REAL AE, BIG, EPS, EWT(*), G, GLAST, H, HMAX, HSIGN,
     8     NROUND, RE, R1MACH, SIZE, SNRM2, SUM, T, TLAST, TOUT, TROOT,
     8     UROUND, WORK(*), Y(*)
      INTEGER IWORK(*)
      LOGICAL CONVRG
      CHARACTER MSG*205
      PARAMETER(NROUND = 20.E0)
      PARAMETER(IAVGH = 1, IHUSED = 2, IAVGRD = 3,
     8          IEL = 4, IH = 160, IHMAX = 161, IHOLD = 162,
     8          IHSIGN = 163, IRC = 164, IRMAX = 165, IT = 166,
     8          ITOUT = 167, ITQ = 168, ITREND = 204, IYH = 205,
     8          INDMXR = 1, INQUSD = 2, INSTEP = 3, INFE = 4, INJE = 5,
     8          INROOT = 6, ICNVRG = 7, IJROOT = 8, IJTASK = 9,
     8          IMNTLD = 10, IMTRLD = 11, INQ = 12, INRTLD = 13,
     8          INDTRT = 14, INWAIT = 15, IMNT = 16, IMTRSV = 17,
     8          IMTR = 18, IMXRDS = 19, IMXORD = 20)
      PARAMETER(INDPRT = 21, INDPVT = 22)
C***FIRST EXECUTABLE STATEMENT  SDRIV3
      NPAR = N
      UROUND = R1MACH (4)
      IF (NROOT .NE. 0) THEN
        AE = R1MACH(1)
```

```
          RE = UROUND
        END IF
        IF (EPS .LT. 0.E0) THEN
          WRITE(MSG, '(''SDRIV36FE Illegal input.  EPS,'', E16.8,
     8    '', is negative.'')') EPS
          CALL XERROR(MSG(1:60), 60, 6, 2)
          RETURN
        END IF
        IF (N .LE. 0) THEN
          WRITE(MSG, '(''SDRIV37FE Illegal input.  Number of equations,'',
     8    I8, '', is not positive.'')') N
          CALL XERROR(MSG(1:72), 72, 7, 2)
          RETURN
        END IF
        IF (MXORD .LE. 0) THEN
          WRITE(MSG, '(''SDRIV314FE Illegal input.  Maximum order,'', I8,
     8    '', is not positive.'')') MXORD
          CALL XERROR(MSG(1:67), 67, 14, 2)
          RETURN
        END IF
        IF ((MINT .LT. 1 .OR. MINT .GT. 3) .OR. (MINT .EQ. 3 .AND.
     8    (MITER .EQ. 0 .OR. MITER .EQ. 3 .OR. IMPL .NE. 0))
     8    .OR. (MITER .LT. 0 .OR. MITER .GT. 5) .OR.
     8    (IMPL .NE. 0 .AND. IMPL .NE. 1 .AND. IMPL .NE. 2) .OR.
     8    ((IMPL .EQ. 1 .OR. IMPL .EQ. 2) .AND. MITER .EQ. 0) .OR.
     8    (IMPL .EQ. 2 .AND. MINT .EQ. 1) .OR.
     8    (NSTATE .LT. 1 .OR. NSTATE .GT. 10)) THEN
          WRITE(MSG, '(''SDRIV39FE Illegal input.  Improper value for '',
     8    ''NSTATE(MSTATE), MINT, MITER or IMPL.'')')
          CALL XERROR(MSG(1:81), 81, 9, 2)
          RETURN
        END IF
        IF (MITER .EQ. 0 .OR. MITER .EQ. 3) THEN
          LIWCHK = INDPVT - 1
        ELSE IF (MITER .EQ. 1 .OR. MITER .EQ. 2 .OR. MITER .EQ. 4 .OR.
     8   MITER .EQ. 5) THEN
          LIWCHK = INDPVT + N - 1
        END IF
        IF (LENIW .LT. LIWCHK) THEN
          WRITE(MSG, '(''SDRIV310FE Illegal input.  Insufficient '',
     8    ''storage allocated for the IWORK array.  Based on the '')')
          WRITE(MSG(94:), '(''value of the input parameters involved, '',
     8    ''the required storage is'', I8)') LIWCHK
          CALL XERROR(MSG(1:164), 164, 10, 2)
          RETURN
        END IF
C                                                    Allocate the WORK array
C                                               IYH is the index of YH in WORK
        IF (MINT .EQ. 1 .OR. MINT .EQ. 3) THEN
          MAXORD = MIN(MXORD, 12)
        ELSE IF (MINT .EQ. 2) THEN
          MAXORD = MIN(MXORD, 5)
        END IF
        IDFDY = IYH + (MAXORD + 1)*N
C                                                  IDFDY is the index of DFDY
C
```

```
      IF (MITER .EQ. 0 .OR. MITER .EQ. 3)  THEN
        IYWT = IDFDY
      ELSE IF (MITER .EQ. 1 .OR. MITER .EQ. 2)  THEN
        IYWT = IDFDY + N*N
      ELSE IF (MITER .EQ. 4 .OR. MITER .EQ. 5)  THEN
        IYWT = IDFDY + (2*ML + MU + 1)*N
      END IF
C                                                IYWT is the index of YWT
      ISAVE1 = IYWT + N
C                                            ISAVE1 is the index of SAVE1
      ISAVE2 = ISAVE1 + N
C                                            ISAVE2 is the index of SAVE2
      IGNOW = ISAVE2 + N
C                                              IGNOW is the index of GNOW
      ITROOT = IGNOW + NROOT
C                                            ITROOT is the index of TROOT
      IFAC = ITROOT + NROOT
C                                                IFAC is the index of FAC
      IF (MITER .EQ. 2 .OR. MITER .EQ. 5 .OR. MINT .EQ. 3) THEN
        IA = IFAC + N
      ELSE
        IA = IFAC
      END IF
C                                                    IA is the index of A
      IF (IMPL .EQ. 0 .OR. MITER .EQ. 3) THEN
        LENCHK = IA - 1
      ELSE IF (IMPL .EQ. 1 .AND. (MITER .EQ. 1 .OR. MITER .EQ. 2)) THEN
        LENCHK = IA - 1 + N*N
      ELSE IF (IMPL .EQ. 1 .AND. (MITER .EQ. 4 .OR. MITER .EQ. 5)) THEN
        LENCHK = IA - 1 + (2*ML + MU + 1)*N
      ELSE IF (IMPL .EQ. 2 .AND. MITER .NE. 3) THEN
        LENCHK = IA - 1 + N
      END IF
      IF (LENW .LT. LENCHK) THEN
        WRITE(MSG, '(''SDRIV38FE Illegal input.  Insufficient '',
     8  ''storage allocated for the WORK array.  Based on the '')')
        WRITE(MSG(92:), '(''value of the input parameters involved, '',
     8  ''the required storage is'', I8)') LENCHK
        CALL XERROR(MSG(1:162), 162, 8, 2)
        RETURN
      END IF
      IF (MITER .EQ. 0 .OR. MITER .EQ. 3) THEN
        MATDIM = 1
      ELSE IF (MITER .EQ. 1 .OR. MITER .EQ. 2) THEN
        MATDIM = N
      ELSE IF (MITER .EQ. 4 .OR. MITER .EQ. 5) THEN
        MATDIM = 2*ML + MU + 1
      END IF
      IF (IMPL .EQ. 0 .OR. IMPL .EQ. 1) THEN
        NDECOM = N
      ELSE IF (IMPL .EQ. 2) THEN
        NDECOM = NDE
      END IF
      IF (NSTATE .EQ. 1) THEN
C                                                   Initialize parameters
        IF (T .EQ. TOUT) RETURN
        IF (MINT .EQ. 1 .OR. MINT .EQ. 3) THEN
          IWORK(IMXORD) = MIN(MXORD, 12)
        ELSE IF (MINT .EQ. 2) THEN
          IWORK(IMXORD) = MIN(MXORD, 5)
        END IF
        IWORK(IMXRDS) = MXORD
        IF (MINT .EQ. 1 .OR. MINT .EQ. 2) THEN
          IWORK(IMNT) = MINT
          IWORK(IMTR) = MITER
          IWORK(IMNTLD) = MINT
          IWORK(IMTRLD) = MITER
        ELSE IF (MINT .EQ. 3) THEN
          IWORK(IMNT) = 1
          IWORK(IMTR) = 0
          IWORK(IMNTLD) = IWORK(IMNT)
          IWORK(IMTRLD) = IWORK(IMTR)
          IWORK(IMTRSV) = MITER
        END IF
        WORK(IHMAX) = HMAX
        H = (TOUT - T)*(1.E0 - 4.E0*UROUND)
        H = SIGN(MIN(ABS(H), HMAX), H)
        WORK(IH) = H
        HSIGN = SIGN(1.E0, H)
        WORK(IHSIGN) = HSIGN
        IWORK(IJTASK) = 0
        WORK(IAVGH) = 0.E0
        WORK(IAVGRD) = 0.E0
        IWORK(INQUSD) = 0
        IWORK(INSTEP) = 0
        IWORK(INFE) = 0
        IWORK(INJE) = 0
        WORK(IT) = T
        IWORK(ICNVRG) = 0
        IWORK(INDPRT) = 0
C                                                 Set initial conditions
        DO 30 I = 1,N
          JYH = I + IYH - 1
 30       WORK(JYH) = Y(I)
        GO TO 180
      END IF
C                                             On a continuation, check
C                                             that output points have
C                                             been or will be overtaken.
      IF (IWORK(ICNVRG) .EQ. 1) THEN
        CONVRG = .TRUE.
      ELSE
        CONVRG = .FALSE.
      END IF
      T = WORK(IT)
      H = WORK(IH)
      HSIGN = WORK(IHSIGN)
      IF (IWORK(IJTASK) .EQ. 0) GO TO 180
C
C                                   IWORK(IJROOT) flags unreported
C                                   roots, and is set to the value of
C                                   NTASK when a root was last selected.
```

```
C                                    It is set to zero when all roots
C                                    have been reported.  IWORK(INROOT)
C                                    contains the index and WORK(ITOUT)
C                                    contains the value of the root last
C                                    selected to be reported.
C                                    IWORK(INRTLD) contains the value of
C                                    NROOT and IWORK(INDTRT) contains
C                                    the value of ITROOT when the array
C                                    of roots was last calculated.
      IF (NROOT .NE. 0) THEN
        JROOT = IWORK(IJROOT)
        IF (JROOT .GT. 0) THEN
C                                       TOUT has just been reported.
C                                       If TROOT .LE. TOUT, report TROOT.
          IF (NSTATE .NE. 5) THEN
            IF (TOUT*HSIGN .GE. WORK(ITOUT)*HSIGN) THEN
              TROOT = WORK(ITOUT)
              CALL SDNTP(H, 0, N, IWORK(INQ), T, TROOT, WORK(IYH),  Y)
              T = TROOT
              NSTATE = 5
              GO TO 580
            END IF
C                                          A root has just been reported.
C                                          Select the next root.
          ELSE
            TROOT = T
            IROOT = 0
            DO 50 I = 1,IWORK(INRTLD)
              JTROOT = IWORK(INDTRT) + I - 1
              IF (WORK(JTROOT)*HSIGN .LE. TROOT*HSIGN) THEN
C
C                                               Check for multiple roots.
C
                IF (WORK(JTROOT) .EQ. WORK(ITOUT) .AND.
     8          I .GT. IWORK(INROOT)) THEN
                  IROOT = I
                  TROOT = WORK(JTROOT)
                  GO TO 60
                END IF
                IF (WORK(JTROOT)*HSIGN .GT. WORK(ITOUT)*HSIGN) THEN
                  IROOT = I
                  TROOT = WORK(JTROOT)
                END IF
              END IF
 50           CONTINUE
 60         IWORK(INROOT) = IROOT
            WORK(ITOUT) = TROOT
            IWORK(IJROOT) = NTASK
            IF (NTASK .EQ. 1) THEN
              IF (IROOT .EQ. 0) THEN
                IWORK(IJROOT) = 0
              ELSE
                IF (TOUT*HSIGN .GE. TROOT*HSIGN) THEN
                  CALL SDNTP(H, 0, N, IWORK(INQ), T, TROOT,WORK(IYH),Y)
                  NSTATE = 5
                  T = TROOT
            GO TO 580
           END IF
          END IF
         ELSE IF (NTASK .EQ. 2 .OR. NTASK .EQ. 3) THEN
C
C                  If there are no more roots, or the
C                  user has altered TOUT to be less
C                  than a root, set IJROOT to zero.
C
          IF (IROOT .EQ. 0 .OR. (TOUT*HSIGN .LT. TROOT*HSIGN)) THEN
           IWORK(IJROOT) = 0
          ELSE
           CALL SDNTP(H, 0, N, IWORK(INQ), T, TROOT, WORK(IYH), Y)
           NSTATE = 5
           T = TROOT
           GO TO 580
          END IF
         END IF
        END IF
       END IF
      END IF
C
      IF (NTASK .EQ. 1) THEN
       NSTATE = 2
       IF (T*HSIGN .GE. TOUT*HSIGN) THEN
        CALL SDNTP (H, 0, N, IWORK(INQ), T, TOUT, WORK(IYH),  Y)
        T = TOUT
        GO TO 580
       END IF
      ELSE IF (NTASK .EQ. 2) THEN
C                           Check if TOUT has
C                           been reset .LT. T
       IF (T*HSIGN .GT. TOUT*HSIGN) THEN
        WRITE(MSG, '(''SDRIV32WRN With NTASK='', I1, '' on input, '',
     8  ''T,'', E16.8, '', was beyond TOUT,'', E16.8, ''.  Solution'',
     8  '' obtained by interpolation.'')') NTASK, T, TOUT
        CALL XERROR(MSG(1:124), 124, 2, 0)
        CALL SDNTP (H, 0, N, IWORK(INQ), T, TOUT, WORK(IYH),  Y)
        T = TOUT
        NSTATE = 2
        GO TO 580
       END IF
C                 Determine if TOUT has been overtaken
C
       IF (ABS(TOUT - T).LE.NROUND*UROUND*MAX(ABS(T), ABS(TOUT))) THEN
        T = TOUT
        NSTATE = 2
        GO TO 560
       END IF
C                      If there are no more roots
C                      to report, report T.
       IF (NSTATE .EQ. 5) THEN
        NSTATE = 2
        GO TO 560
       END IF
       NSTATE = 2
```

```
C                                                            See if TOUT will
C                                                            be overtaken.
        IF ((T + H)*HSIGN .GT. TOUT*HSIGN) THEN
          H = TOUT - T
          IF ((T + H)*HSIGN .GT. TOUT*HSIGN) H = H*(1.E0 - 4.E0*UROUND)
          WORK(IH) = H
          IF (H .EQ. 0.E0) GO TO 670
          IWORK(IJTASK) = -1
        END IF
      ELSE IF (NTASK .EQ. 3) THEN
        NSTATE = 2
        IF (T*HSIGN .GT. TOUT*HSIGN) THEN
          WRITE(MSG, '(''SDRIV32WRN With NTASK='', I1, '' on input, '',
     8    ''T,'', E16.8, '', was beyond TOUT,'', E16.8, ''.  Solution'',
     8    '' obtained by interpolation.'')') NTASK, T, TOUT
          CALL XERROR(MSG(1:124), 124, 2, 0)
          CALL SDNTP (H, 0, N, IWORK(INQ), T, TOUT, WORK(IYH),  Y)
          T = TOUT
          GO TO 580
        END IF
        IF (ABS(TOUT - T).LE.NROUND*UROUND*MAX(ABS(T), ABS(TOUT))) THEN
          T = TOUT
          GO TO 560
        END IF
        IF ((T + H)*HSIGN .GT. TOUT*HSIGN) THEN
          H = TOUT - T
          IF ((T + H)*HSIGN .GT. TOUT*HSIGN) H = H*(1.E0 - 4.E0*UROUND)
          WORK(IH) = H
          IF (H .EQ. 0.E0) GO TO 670
          IWORK(IJTASK) = -1
        END IF
      END IF
C                               Implement changes in MINT, MITER, and/or HMAX.
C
      IF ((MINT .NE. IWORK(IMNTLD) .OR. MITER .NE. IWORK(IMTRLD)) .AND.
     8  MINT .NE. 3 .AND. IWORK(IMNTLD) .NE. 3) IWORK(IJTASK) = -1
      IF (HMAX .NE. WORK(IHMAX)) THEN
        H = SIGN(MIN(ABS(H), HMAX), H)
        IF (H .NE. WORK(IH)) THEN
          IWORK(IJTASK) = -1
          WORK(IH) = H
        END IF
        WORK(IHMAX) = HMAX
      END IF
C
 180  NSTEPL = IWORK(INSTEP)
      DO 190 I = 1,N
        JYH = IYH + I - 1
 190    Y(I) = WORK(JYH)
      IF (NROOT .NE. 0) THEN
        DO 200 I = 1,NROOT
          JGNOW = IGNOW + I - 1
          WORK(JGNOW) = G (NPAR, T, Y, I)
          IF (NPAR .EQ. 0) THEN
            IWORK(INROOT) = I
            NSTATE = 7
          RETURN
        END IF
 200    CONTINUE
      END IF
      IF (IERROR .EQ. 1) THEN
        DO 230 I = 1,N
          JYWT = I + IYWT - 1
 230      WORK(JYWT) = 1.E0
        GO TO 410
      ELSE IF (IERROR .EQ. 5) THEN
        DO 250 I = 1,N
          JYWT = I + IYWT - 1
 250      WORK(JYWT) = EWT(I)
        GO TO 410
      END IF
C                                     Reset YWT array.  Looping point.
 260  IF (IERROR .EQ. 2) THEN
        DO 280 I = 1,N
          IF (Y(I) .EQ. 0.E0) GO TO 290
          JYWT = I + IYWT - 1
 280      WORK(JYWT) = ABS(Y(I))
        GO TO 410
 290    IF (IWORK(IJTASK) .EQ. 0) THEN
          CALL F (NPAR, T, Y, WORK(ISAVE2))
          IF (NPAR .EQ. 0) THEN
            NSTATE = 6
            RETURN
          END IF
          IWORK(INFE) = IWORK(INFE) + 1
          IF (MITER .EQ. 3 .AND. IMPL .NE. 0) THEN
            IFLAG = 0
            CALL USERS(Y, WORK(IYH), WORK(IYWT), WORK(ISAVE1),
     8                 WORK(ISAVE2), T, H, WORK(IEL), IMPL, NPAR,
     8                 NDECOM, IFLAG)
            IF (NPAR .EQ. 0) THEN
              NSTATE = 10
              RETURN
            END IF
          ELSE IF (IMPL .EQ. 1) THEN
            IF (MITER .EQ. 1 .OR. MITER .EQ. 2) THEN
              CALL FA (NPAR, T, Y, WORK(IA), MATDIM, ML, MU, NDECOM)
              IF (NPAR .EQ. 0) THEN
                NSTATE = 9
                RETURN
              END IF
              CALL SGEFA (WORK(IA), MATDIM, N, IWORK(INDPVT), INFO)
              IF (INFO .NE. 0) GO TO 690
              CALL SGESL(WORK(IA),MATDIM,N,IWORK(INDPVT),WORK(ISAVE2),0)
            ELSE IF (MITER .EQ. 4 .OR. MITER .EQ. 5) THEN
              JAML = IA + ML
              CALL FA (NPAR, T, Y, WORK(JAML), MATDIM, ML, MU, NDECOM)
              IF (NPAR .EQ. 0) THEN
                NSTATE = 9
                RETURN
              END IF
              CALL SGBFA (WORK(IA),MATDIM,N,ML,MU,IWORK(INDPVT),INFO)
              IF (INFO .NE. 0) GO TO 690
              CALL SGBSL (WORK(IA), MATDIM, N, ML, MU, IWORK(INDPVT),
     8                    WORK(ISAVE2), 0)
            END IF
          ELSE IF (IMPL .EQ. 2) THEN
            CALL FA (NPAR, T, Y, WORK(IA), MATDIM, ML, MU, NDECOM)
            IF (NPAR .EQ. 0) THEN
              NSTATE = 9
              RETURN
            END IF
            DO 340 I = 1,NDECOM
              JA = I + IA - 1
              JSAVE2 = I + ISAVE2 - 1
              IF (WORK(JA) .EQ. 0.E0) GO TO 690
 340          WORK(JSAVE2) = WORK(JSAVE2)/WORK(JA)
          END IF
        END IF
        DO 360 J = I,N
          JYWT = J + IYWT - 1
          IF (Y(J) .NE. 0.E0) THEN
            WORK(JYWT) = ABS(Y(J))
          ELSE
            IF (IWORK(IJTASK) .EQ. 0) THEN
              JSAVE2 = J + ISAVE2 - 1
              WORK(JYWT) = ABS(H*WORK(JSAVE2))
            ELSE
              JHYP = J + IYH + N - 1
              WORK(JYWT) = ABS(WORK(JHYP))
            END IF
          END IF
          IF (WORK(JYWT) .EQ. 0.E0) WORK(JYWT) = UROUND
 360      CONTINUE
      ELSE IF (IERROR .EQ. 3) THL
        DO 380 I = 1,N
          JYWT = I + IYWT - 1
 380      WORK(JYWT) = MAX(EWT(1), ABS(Y(I)))
      ELSE IF (IERROR .EQ. 4) THEN
        DO 400 I = 1,N
          JYWT = I + IYWT - 1
 400      WORK(JYWT) = MAX(EWT(I), ABS(Y(I)))
      END IF
C
 410  DO 420 I = 1,N
        JYWT = I + IYWT - 1
        JSAVE2 = I + ISAVE2 - 1
 420    WORK(JSAVE2) = Y(I)/WORK(JYWT)
      SUM = SNRM2(N, WORK(ISAVE2), 1)/SQRT(REAL(N))
      IF (EPS .LT. SUM*UROUND) THEN
        EPS = SUM*UROUND*(1.E0 + 10.E0*UROUND)
        WRITE(MSG, '(''SDRIV34REC At T,'', E16.8, '', the requested '',
     8  ''accuracy, EPS, was not obtainable with the machine '',
     8  ''precision.  EPS has been increased to'')') T
        WRITE(MSG(137:), '(E16.8)') EPS
        CALL XERROR(MSG(1:152), 152, 4, 1)
        NSTATE = 4
        GO TO 560
      END IF
      IF (ABS(H) .GE. UROUND*ABS(T)) THEN
        IWORK(INDPRT) = 0
      ELSE IF (IWORK(INDPRT) .EQ. 0) THEN
        WRITE(MSG, '(''SDRIV35WRN At T,'', E16.8, '', the step size,'',
     8  E16.8, '', is smaller than the roundoff level of T.  '')') T, H
        WRITE(MSG(109:), '(''This may occur if there is an abrupt '',
     8  ''change in the right hand side of the differential '',
     8  ''equations.'')')
        CALL XERROR(MSG(1:205), 205, 5, 0)
        IWORK(INDPRT) = 1
      END IF
      IF (NTASK.NE.2) THEN
        IF ((IWORK(INSTEP)-NSTEPL) .GT. MXSTEP) THEN
          WRITE(MSG, '(''SDRIV33WRN At T,'', E16.8, '', '', I3,
     8    '' steps have been taken without reaching TOUT,'', E16.8)')
     8    T, MXSTEP, TOUT
          CALL XERROR(MSG(1:103), 103, 3, 0)
          NSTATE = 3
          GO TO 560
        END IF
      END IF
C
C     CALL SDSTP (EPS, F, FA, HMAX, IMPL, JACOBN, MATDIM, MAXORD,
C    8            MINT, MITER, ML, MU, N, NDE, YWT, UROUND, USERS,
C    8            AVGH, AVGORD, H, HUSED, JTASK, MNTOLD, MTROLD,
C    8            NFE, NJE, NQUSED, NSTEP, T, Y, YH,  A, CONVRG,
C    8            DFDY, EL, FAC, HOLD, IPVT, JSTATE, NQ, NWAIT, RC,
C    8            RMAX, SAVE1, SAVE2, TQ, TREND, ISWFLG, MTRSV, MXRDSV)
C
      CALL SDSTP (EPS, F, FA, WORK(IHMAX), IMPL, JACOBN, MATDIM,
     8            IWORK(IMXORD), IWORK(IMNT), IWORK(IMTR), ML, MU, NPAR,
     8           NDECOM, WORK(IYWT), UROUND, USERS,  WORK(IAVGH),
     8           WORK(IAVGRD), WORK(IH), WORK(IHUSED), IWORK(IJTASK),
     8           IWORK(IMNTLD), IWORK(IMTRLD), IWORK(INFE), IWORK(INJE),
     8            IWORK(INQUSD), IWORK(INSTEP), WORK(IT), Y, WORK(IYH),
     8            WORK(IA), CONVRG, WORK(IDFDY), WORK(IEL), WORK(IFAC),
     8            WORK(IHOLD), IWORK(INDPVT), JSTATE, IWORK(INQ),
     8            IWORK(INWAIT), WORK(IRC), WORK(IRMAX), WORK(ISAVE1),
     8            WORK(ISAVE2), WORK(ITQ), WORK(ITREND), MINT,
     8            IWORK(IMTRSV), IWORK(IMXRDS))
      T = WORK(IT)
      H = WORK(IH)
      GO TO (470, 670, 680, 690, 690, 660, 660, 660, 660, 660), JSTATE
 470  IWORK(IJTASK) = 1
C                                 Determine if a root has been overtaken
      IF (NROOT .NE. 0) THEN
        IROOT = 0
        DO 500 I = 1,NROOT
          JTROOT = ITROOT + I - 1
          JGNOW = IGNOW + I - 1
          GLAST = WORK(JGNOW)
          WORK(JGNOW) = G (NPAR, T, Y, I)
          IF (NPAR .EQ. 0) THEN
            IWORK(INROOT) = I
            NSTATE = 7
      END IF
      IF (ABS(H) .GE. UROUND*ABS(T)) THEN
        IWORK(INDPRT) = 0
      ELSE IF (IWORK(INDPRT) .EQ. 0) THEN
        WRITE(MSG, '(''SDRIV35WRN At T,'', E16.8, '', the step size,'',
     8  E16.8, '', is smaller than the roundoff level of T.  '')') T, H
        WRITE(MSG(109:), '(''This may occur if there is an abrupt '',
     8  ''change in the right hand side of the differential '',
     8  ''equations.'')')
        CALL XERROR(MSG(1:205), 205, 5, 0)
        IWORK(INDPRT) = 1
      END IF
      IF (NTASK.NE.2) THEN
        IF ((IWORK(INSTEP)-NSTEPL) .GT. MXSTEP) THEN
          WRITE(MSG, '(''SDRIV33WRN At T,'', E16.8, '', '', I8,
     8    '' steps have been taken without reaching TOUT,'', E16.8)')
     8    T, MXSTEP, TOUT
          CALL XERROR(MSG(1:103), 103, 3, 0)
          NSTATE = 3
          GO TO 560
        END IF
      END IF
C
C     CALL SDSTP (EPS, F, FA, HMAX, IMPL, JACOBN, MATDIM, MAXORD,
C    8            MINT, MITER, ML, MU, N, NDE, YWT, UROUND, USERS,
C    8            AVGH, AVGORD, H, HUSED, JTASK, MNTOLD, MTROLD,
C    8            NFE, NJE, NQUSED, NSTEP, T, Y, YH,  A, CONVRG,
C    8            DFDY, EL, FAC, HOLD, IPVT, JSTATE, NQ, NWAIT, RC,
C    8            RMAX, SAVE1, SAVE2, TQ, TREND, ISWFLG, MTRSV, MXRDSV)
C
      CALL SDSTP (EPS, F, FA, WORK(IHMAX), IMPL, JACOBN, MATDIM,
     8            IWORK(IMXORD), IWORK(IMNT), IWORK(IMTR), ML, MU, NPAR,
     8           NDECOM, WORK(IYWT), UROUND, USERS,  WORK(IAVGH),
     8           WORK(IAVGRD), WORK(IH), WORK(IHUSED), IWORK(IJTASK),
     8           IWORK(IMNTLD), IWORK(IMTRLD), IWORK(INFE), IWORK(INJE),
     8            IWORK(INQUSD), IWORK(INSTEP), WORK(IT), Y, WORK(IYH),
     8            WORK(IA), CONVRG, WORK(IDFDY), WORK(IEL), WORK(IFAC),
     8            WORK(IHOLD), IWORK(INDPVT), JSTATE, IWORK(INQ),
     8            IWORK(INWAIT), WORK(IRC), WORK(IRMAX), WORK(ISAVE1),
     8            WORK(ISAVE2), WORK(ITQ), WORK(ITREND), MINT,
     8            IWORK(IMTRSV), IWORK(IMXRDS))
      T = WORK(IT)
      H = WORK(IH)
      GO TO (470, 670, 680, 690, 690, 660, 660, 660, 660, 660), JSTATE
 470  IWORK(IJTASK) = 1
C                                 Determine if a root has been overtaken
      IF (NROOT .NE. 0) THEN
        IROOT = 0
        DO 500 I = 1,NROOT
          JTROOT = ITROOT + I - 1
          JGNOW = IGNOW + I - 1
          GLAST = WORK(JGNOW)
          WORK(JGNOW) = G (NPAR, T, Y, I)
          IF (NPAR .EQ. 0) THEN
            IWORK(INROOT) = I
            NSTATE = 7
        END IF
      END IF
C                               Test for NTASK condition to be satisfied
      NSTATE = 2
      IF (NTASK .EQ. 1) THEN
        IF (T*HSIGN .LT. TOUT*HSIGN) GO TO 260
        CALL SDNTP (H, 0, N, IWORK(INQ), T, TOUT, WORK(IYH),  Y)
        T = TOUT
        GO TO 580
C                               TOUT is assumed to have been attained
C                               exactly if T is within twenty roundoff
C                               units of TOUT, relative to max(TOUT, T).
      ELSE IF (NTASK .EQ. 2) THEN
        IF (ABS(TOUT - T).LE.NROUND*UROUND*MAX(ABS(T), ABS(TOUT))) THEN
          T = TOUT
        ELSE
          IF ((T + H)*HSIGN .GT. TOUT*HSIGN) THEN
            H = TOUT - T
            IF ((T + H)*HSIGN.GT.TOUT*HSIGN) H = H*(1.E0 - 4.E0*UROUND)
            WORK(IH) = H
            IF (H .EQ. 0.E0) GO TO 670
            IWORK(IJTASK) = -1
          END IF
        END IF
      ELSE IF (NTASK .EQ. 3) THEN
        IF (ABS(TOUT - T).LE.NROUND*UROUND*MAX(ABS(T), ABS(TOUT))) THEN
          T = TOUT
        ELSE
          IF ((T + H)*HSIGN .GT. TOUT*HSIGN) THEN
            H = TOUT - T
            IF ((T + H)*HSIGN.GT.TOUT*HSIGN) H = H*(1.E0 - 4.E0*UROUND)
            WORK(IH) = H
            IF (H .EQ. 0.E0) GO TO 670
            IWORK(IJTASK) = -1
          END IF
          GO TO 260
        END IF
      END IF
C                                      All returns are made through this
C                                      section.  IMXERR is determined.
 560  DO 570 I = 1,N
        JYH = I + IYH - 1
 570    Y(I) = WORK(JYH)
 580  IF (CONVRG) THEN
        IWORK(ICNVRG) = 1
      ELSE
        IWORK(ICNVRG) = 0
      END IF
      IF (IWORK(IJTASK) .EQ. 0) RETURN
      BIG = 0.E0
      IMXERR = 1
      IWORK(INDMXR) = IMXERR
      DO  590 I = 1,N
C                                            SIZE = ABS(ERROR(I)/YWT(I))
        JYWT = I + IYWT - 1
        JERROR = I + ISAVE1 - 1
```

```
          SIZE = ABS(WORK(JERROR)/WORK(JYWT))
          IF (BIG .LT. SIZE) THEN
            BIG = SIZE
            IMXERR = I
            IWORK(INDMXR) = IMXERR
          END IF
 590      CONTINUE
        RETURN
C
 660  NSTATE = JSTATE
      RETURN
C                                              Fatal errors are processed here
C
 670  WRITE(MSG, '(''SDRIV311FE At T,'', E16.8, '', the attempted '',
     8  ''step size has gone to zero.  Often this occurs if the '',
     8  ''problem setup is incorrect.'')') T
      CALL XERROR(MSG(1:129), 129, 11, 2)
      RETURN
C
 680  WRITE(MSG, '(''SDRIV312FE At T,'', E16.8, '', the step size has'',
     8  '' been reduced about 50 times without advancing the '')') T
      WRITE(MSG(103:), '(''solution.  Often this occurs if the '',
     8  ''problem setup is incorrect.'')')
      CALL XERROR(MSG(1:165), 165, 12, 2)
      RETURN
C
 690  WRITE(MSG, '(''SDRIV313FE At T,'', E16.8, '', while solving'',
     8  '' A*YDOT = F, A is singular.'')') T
      CALL XERROR(MSG(1:74), 74, 13, 2)
      RETURN
      END
      SUBROUTINE SDSCL (HMAX,N,NQ,RMAX,H,RC,RH,YH)
C***BEGIN PROLOGUE  SDSCL
C***REFER TO  SDRIV3
C   This subroutine rescales the YH array whenever the step size
C   is changed.
C***ROUTINES CALLED  (NONE)
C***DATE WRITTEN   790601   (YYMMDD)
C***REVISION DATE  850319   (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C           SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDSCL
      REAL H, HMAX, RC, RH, RMAX, R1, YH(N,*)
C***FIRST EXECUTABLE STATEMENT  SDSCL
      IF (H .LT. 1.E0) THEN
        RH = MIN(ABS(H)*RH, ABS(H)*RMAX, HMAX)/ABS(H)
      ELSE
        RH = MIN(RH, RMAX, HMAX/ABS(H))
      END IF
      R1 = 1.E0
      DO 10 J = 1,NQ
        R1 = R1*RH
        DO 10 I = 1,N
 10       YH(I,J+1) = YH(I,J+1)*R1
      H = H*RH
      RC = RC*RH
      END
      SUBROUTINE SDSTP (EPS,F,FA,HMAX,IMPL,JACOBN,MATDIM,MAXORD,MINT,
     8   MITER,ML,MU,N,NDE,YWT,UROUND,USERS,AVGH,AVGORD,H,HUSED,JTASK,
     8   MNTOLD,MTROLD,NFE,NJE,NQUSED,NSTEP,T,Y,YH,A,CONVRG,DFDY,EL,FAC,
     8   HOLD,IPVT,JSTATE,NQ,NWAIT,RC,RMAX,SAVE1,SAVE2,TQ,TREND,ISWFLG,
     8   MTRSV,MXRDSV)
C***BEGIN PROLOGUE  SDSTP
C***REFER TO  SDRIV3
C  SDSTP performs one step of the integration of an initial value
C  problem for a system of ordinary differential equations.
C***ROUTINES CALLED  SDNTL,SDPST,SDCOR,SDPSC,SDSCL,SNRM2
C***DATE WRITTEN   790601   (YYMMDD)
C***REVISION DATE  870810   (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C           SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDSTP
      EXTERNAL F, JACOBN, FA, USERS
      REAL A(MATDIM,*), AVGH, AVGORD, BIAS1, BIAS2, BIAS3,
     8     BND, CTEST, D, DENOM, DFDY(MATDIM,*), D1, EL(13,12), EPS,
     8     ERDN, ERUP, ETEST, FAC(*), H, HMAX, HN, HOLD, HS, HUSED,
     8     NUMER, RC, RCTEST, RH, RH1, RH2, RH3, RMAX, RMFAIL, RMNORM,
     8     SAVE1(*), SAVE2(*), SNRM2, T, TOLD, TQ(3,12), TREND, TRSHLD,
     8     UROUND, Y(*), YH(N,*), YWT(*), YONRM
      INTEGER IPVT(*)
      LOGICAL CONVRG, EVALFA, EVALJC, IER, SWITCH
      PARAMETER(BIAS1 = 1.3E0, BIAS2 = 1.2E0, BIAS3 = 1.4E0, MXFAIL = 3,
     8          MXITER = 3, MXTRY = 50, RCTEST = .3E0, RMFAIL = 2.E0,
     8          RMNORM = 10.E0, TRSHLD = 1.E0)
      DATA IER /.FALSE./
C***FIRST EXECUTABLE STATEMENT  SDSTP
      NSV = N
      BND = 0.E0
      SWITCH = .FALSE.
      NTRY = 0
      TOLD = T
      NFAIL = 0
      IF (JTASK .LE. 0) THEN
        CALL SDNTL (EPS, F, FA, HMAX, HOLD, IMPL, JTASK, MATDIM,
     8              MAXORD, MINT, MITER, ML, MU, N, NDE, SAVE1, T,
     8              UROUND, USERS, Y, YWT,  H, MNTOLD, MTROLD, NFE, RC,
     8              YH,  A, CONVRG, EL, FAC, IER, IPVT, NQ, NWAIT, RH,
     8              RMAX, SAVE2, TQ, TREND, ISWFLG, JSTATE)
        IF (N .EQ. 0) GO TO 440
        IF (H .EQ. 0.E0) GO TO 400
        IF (IER) GO TO 420
      END IF
 100  NTRY = NTRY + 1
      IF (NTRY .GT. MXTRY) GO TO 410
      T = T + H
      CALL SDPSC (1, N, NQ,  YH)
      EVALJC = ((ABS(RC - 1.E0) .GT. RCTEST) .AND. (MITER .NE. 0))
      EVALFA = .NOT. EVALJC
C
 110  ITER = 0
```

```
      DO 115 I = 1,N
 115    Y(I) = YH(I,1)
      CALL F (N, T, Y, SAVE2)
      IF (N .EQ. 0) THEN
        JSTATE = 6
        GO TO 430
      END IF
      NFE = NFE + 1
      IF (EVALJC .OR. IER) THEN
        CALL SDPST (EL, F, FA, H, IMPL, JACOBN, MATDIM, MITER, ML,
     8              MU, N, NDE, NQ, SAVE2, T, USERS, Y, YH, YWT, UROUND,
     8              NFE, NJE,  A, DFDY, FAC, IER, IPVT, SAVE1, ISWFLG,
     8              BND, JSTATE)
        IF (N .EQ. 0) GO TO 430
        IF (IER) GO TO 160
        CONVRG = .FALSE.
        RC = 1.E0
      END IF
      DO 125 I = 1,N
 125    SAVE1(I) = 0.E0
C                         Up to MXITER corrector iterations are taken.
C                         Convergence is tested by requiring the r.m.s.
C                         norm of changes to be less than EPS.  The sum of
C                         the corrections is accumulated in the vector
C                         SAVE1(I).  It is approximately equal to the L-th
C                         derivative of Y multiplied by
C                         H**L/(factorial(L-1)*EL(L,NQ)), and is thus
C                         proportional to the actual errors to the lowest
C                         power of H present (H**L).  The YH array is not
C                         altered in the correction loop.  The norm of the
C                         iterate difference is stored in D.  If
C                         ITER .GT. 0, an estimate of the convergence rate
C                         constant is stored in TREND, and this is used in
C                         the convergence test.
C
 130  CALL SDCOR (DFDY, EL, FA, H, IMPL, IPVT, MATDIM, MITER, ML,
     8            MU, N, NDE, NQ, T, USERS, Y, YH, YWT,  EVALFA, SAVE1,
     8            SAVE2,  A, D, JSTATE)
        IF (N .EQ. 0) GO TO 430
      IF (ISWFLG .EQ. 3 .AND. MINT .EQ. 1) THEN
        IF (ITER .EQ. 0) THEN
          NUMER = SNRM2(N, SAVE1, 1)
          DO 132 I = 1,N
 132        DFDY(1,I) = SAVE1(I)
          YONRM = SNRM2(N, YH, 1)
        ELSE
          DENOM = NUMER
          DO 134 I = 1,N
 134        DFDY(1,I) = SAVE1(I) - DFDY(1,I)
          NUMER = SNRM2(N, DFDY, MATDIM)
          IF (EL(1,NQ)*NUMER .LE. 100.E0*UROUND*YONRM) THEN
            IF (RMAX .EQ. RMFAIL) THEN
              SWITCH = .TRUE.
              GO TO 170
            END IF
          END IF
```

```
         DO 136 I = 1,N
 136       DFDY(1,I) = SAVE1(I)
         IF (DENOM .NE. 0.E0)
     8   BND = MAX(BND, NUMER/(DENOM*ABS(H)*EL(1,NQ)))
       END IF
     END IF
     IF (ITER .GT. 0) TREND = MAX(.9E0*TREND, D/D1)
     D1 = D
     CTEST = MIN(2.E0*TREND, 1.E0)*D
     IF (CTEST .LE. EPS) GO TO 170
     ITER = ITER + 1
     IF (ITER .LT. MXITER) THEN
       DO 140 I = 1,N
 140     Y(I) = YH(I,1) + EL(1,NQ)*SAVE1(I)
       CALL F (N, T, Y, SAVE2)
       IF (N .EQ. 0) THEN
         JSTATE = 6
         GO TO 430
       END IF
       NFE = NFE + 1
       GO TO 130
     END IF
C                     The corrector iteration failed to converge in
C                     MXITER tries.  If partials are involved but are
C                     not up to date, they are reevaluated for the next
C                     try.  Otherwise the YH array is retracted to its
C                     values before prediction, and H is reduced, if
C                     possible.  If not, a no-convergence exit is taken.
     IF (CONVRG) THEN
       EVALJC = .TRUE.
       EVALFA = .FALSE.
       GO TO 110
     END IF
 160 T = TOLD
     CALL SDPSC (-1, N, NQ,  YH)
     NWAIT = NQ + 2
     IF (JTASK .NE. 0 .AND. JTASK .NE. 2) RMAX = RMFAIL
     IF (ITER .EQ. 0) THEN
       RH = .3E0
     ELSE
       RH = .9E0*(EPS/CTEST)**(.2E0)
     END IF
     IF (RH*H .EQ. 0.E0) GO TO 400
     CALL SDSCL (HMAX, N, NQ, RMAX,  H, RC, RH, YH)
     GO TO 100
C                          The corrector has converged.  CONVRG is set
C                          to .TRUE. if partial derivatives were used,
C                          to indicate that they may need updating on
C                          subsequent steps.  The error test is made.
 170 CONVRG = (MITER .NE. 0)
     DO 180 I = 1,NDE
 180   SAVE2(I) = SAVE1(I)/YWT(I)
     ETEST = SNRM2(NDE, SAVE2, 1)/(TQ(2,NQ)*SQRT(REAL(NDE)))
C
C                           The error test failed.  NFAIL keeps track of
C                           multiple failures.  Restore T and the YH
```

```
C                             array to their previous values, and prepare
C                             to try the step again.  Compute the optimum
C                             step size for this or one lower order.
      IF (ETEST .GT. EPS) THEN
        T = TOLD
        CALL SDPSC (-1, N, NQ,  YH)
        NFAIL = NFAIL + 1
        IF (NFAIL .LT. MXFAIL) THEN
          IF (JTASK .NE. 0 .AND. JTASK .NE. 2) RMAX = RMFAIL
          RH2 = 1.E0/(BIAS2*(ETEST/EPS)**(1.E0/REAL(NQ+1)))
          IF (NQ .GT. 1) THEN
            DO 190 I = 1,NDE
 190          SAVE2(I) = YH(I,NQ+1)/YWT(I)
            ERDN = SNRM2(NDE, SAVE2, 1)/(TQ(1,NQ)*SQRT(REAL(NDE)))
            RH1 = 1.E0/MAX(1.E0, BIAS1*(ERDN/EPS)**(1.E0/REAL(NQ)))
            IF (RH2 .LT. RH1) THEN
              NQ = NQ - 1
              RC = RC*EL(1,NQ)/EL(1,NQ+1)
              RH = RH1
            ELSE
              RH = RH2
            END IF
          ELSE
            RH = RH2
          END IF
          NWAIT = NQ + 2
          IF (RH*H .EQ. 0.E0) GO TO 400
          CALL SDSCL (HMAX, N, NQ, RMAX,  H, RC, RH, YH)
          GO TO 100
        END IF
C                 Control reaches this section if the error test has
C                 failed MXFAIL or more times.  It is assumed that the
C                 derivatives that have accumulated in the YH array have
C                 errors of the wrong order.  Hence the first derivative
C                 is recomputed, the order is set to 1, and the step is
C                 retried.
        NFAIL = 0
        JTASK = 2
        DO 215 I = 1,N
 215      Y(I) = YH(I,1)
        CALL SDNTL (EPS, F, FA, HMAX, HOLD, IMPL, JTASK, MATDIM,
     8              MAXORD, MINT, MITER, ML, MU, N, NDE, SAVE1, T,
     8              UROUND, USERS, Y, YWT,  H, MNTOLD, MTROLD, NFE, RC,
     8              YH,  A, CONVRG, EL, FAC, IER, IPVT, NQ, NWAIT, RH,
     8              RMAX, SAVE2, TQ, TREND, ISWFLG, JSTATE)
        RMAX = RMNORM
        IF (N .EQ. 0) GO TO 440
        IF (H .EQ. 0.E0) GO TO 400
        IF (IER) GO TO 420
        GO TO 100
      END IF
C                             After a successful step, update the YH array.
      NSTEP = NSTEP + 1
      HUSED = H
      NQUSED = NQ
      AVGH = (REAL(NSTEP-1)*AVGH + H)/REAL(NSTEP)
      AVGORD - (REAL(NSTEP-1)*AVGORD + REAL(NQ))/REAL(NSTEP)
      DO 230 J - 1,NQ+1
        DO 230 I - 1,N
 230      YH(I,J) - YH(I,J) + EL(J,NQ)*SAVE1(I)
      DO 235 I - 1,N
 235    Y(I) - YH(I,1)
C                                       If ISWFLG is 3, consider
C                                       changing integration methods.
C
      IF (ISWFLG .EQ. 3) THEN
        IF (BND .NE. 0.E0) THEN
          IF (MINT .EQ. 1 .AND. NQ .LE. 5) THEN
            HN - ABS(H)/MAX(UROUND, (ETEST/EPS)**(1.E0/REAL(NQ+1)))
            HN - MIN(HN, 1.E0/(2.E0*EL(1,NQ)*BND))
            HS - ABS(H)/MAX(UROUND,
     8      (ETEST/(EPS*EL(NQ+1,1)))**(1.E0/REAL(NQ+1)))
            IF (HS .GT. 1.2E0*HN) THEN
              MINT - 2
              MNTOLD - MINT
              MITER - MTRSV
              MTROLD - MITER
              MAXORD - MIN(MXRDSV, 5)
              RC - 0.E0
              RMAX - RMNORM
              TREND - 1.E0
              CALL SDCST (MAXORD, MINT, ISWFLG, EL, TQ)
              NWAIT - NQ + 2
            END IF
          ELSE IF (MINT .EQ. 2) THEN
            HS - ABS(H)/MAX(UROUND, (ETEST/EPS)**(1.E0/REAL(NQ+1)))
            HN - ABS(H)/MAX(UROUND,
     8      (ETEST*EL(NQ+1,1)/EPS)**(1.E0/REAL(NQ+1)))
            HN - MIN(HN, 1.E0/(2.E0*EL(1,NQ)*BND))
            IF (HN .GE. HS) THEN
              MINT - 1
              MNTOLD - MINT
              MITER - 0
              MTROLD - MITER
              MAXORD - MIN(MXRDSV, 12)
              RMAX - RMNORM
              TREND - 1.E0
              CONVRG - .FALSE.
              CALL SDCST (MAXORD, MINT, ISWFLG, EL, TQ)
              NWAIT - NQ + 2
            END IF
          END IF
        END IF
      END IF
      IF (SWITCH) THEN
        MINT - 2
        MNTOLD - MINT
        MITER - MTRSV
        MTROLD - MITER
        MAXORD - MIN(MXRDSV, 5)
        NQ - MIN(NQ, MAXORD)
        RC - 0.E0
```

```
       RMAX = RMNORM
       TREND = 1.E0
       CALL SDCST (MAXORD, MINT, ISWFLG, EL, TQ)
       NWAIT = NQ + 2
      END IF
C                  Consider changing H if NWAIT = 1.  Otherwise
C                  decrease NWAIT by 1.  If NWAIT is then 1 and
C                  NQ.LT.MAXORD, then SAVE1 is saved for use in
C                  a possible order increase on the next step.
C
      IF (JTASK .EQ. 0 .OR. JTASK .EQ. 2) THEN
       RH = 1.E0/MAX(UROUND, BIAS2*(ETEST/EPS)**(1.E0/REAL(NQ+1)))
       IF (RH .GT. TRSHLD) CALL SDSCL (HMAX, N, NQ, RMAX, H, RC, RH, YH)
      ELSE IF (NWAIT .GT. 1) THEN
       NWAIT = NWAIT - 1
       IF (NWAIT .EQ. 1 .AND. NQ .LT. MAXORD) THEN
        DO 250 I = 1,NDE
 250     YH(I,MAXORD+1) = SAVE1(I)
       END IF
C       If a change in H is considered, an increase or decrease in
C       order by one is considered also.  A change in H is made
C       only if it is by a factor of at least TRSHLD.  Factors
C       RH1, RH2, and RH3 are computed, by which H could be
C       multiplied at order NQ - 1, order NQ, or order NQ + 1,
C       respectively.  The largest of these is determined and the
C       new order chosen accordingly.  If the order is to be
C       increased, we compute one additional scaled derivative.
C       If there is a change of order, reset NQ and the
C       coefficients.  In any case H is reset according to RH and
C       the YH array is rescaled.
      ELSE
       IF (NQ .EQ. 1) THEN
        RH1 = 0.E0
       ELSE
        DO 270 I = 1,NDE
 270     SAVE2(I) = YH(I,NQ+1)/YWT(I)
        ERDN = SNRM2(NDE, SAVE2, 1)/(TQ(1,NQ)*SQRT(REAL(NDE)))
        RH1 = 1.E0/MAX(UROUND, BIAS1*(ERDN/EPS)**(1.E0/REAL(NQ)))
       END IF
       RH2 = 1.E0/MAX(UROUND, BIAS2*(ETEST/EPS)**(1.E0/REAL(NQ+1)))
       IF (NQ .EQ. MAXORD) THEN
        RH3 = 0.E0
       ELSE
        DO 290 I = 1,NDE
 290     SAVE2(I) = (SAVE1(I) - YH(I,MAXORD+1))/YWT(I)
        ERUP = SNRM2(NDE, SAVE2, 1)/(TQ(3,NQ)*SQRT(REAL(NDE)))
        RH3 = 1.E0/MAX(UROUND, BIAS3*(ERUP/EPS)**(1.E0/REAL(NQ+2)))
       END IF
       IF (RH1 .GT. RH2 .AND. RH1 .GE. RH3) THEN
        RH = RH1
        IF (RH .LE. TRSHLD) GO TO 380
        NQ = NQ - 1
        RC = RC*EL(1,NQ)/EL(1,NQ+1)
       ELSE IF (RH2 .GE. RH1 .AND. RH2 .GE. RH3) THEN
        RH = RH2
        IF (RH .LE. TRSHLD) GO TO 380
```

```
          ELSE
            RH = RH3
            IF (RH .LE. TRSHLD) GO TO 380
            DO 360 I = 1,N
 360          YH(I,NQ+2) = SAVE1(I)*EL(NQ+1,NQ)/REAL(NQ+1)
            NQ = NQ + 1
            RC = RC*EL(1,NQ)/EL(1,NQ-1)
          END IF
          IF (ISWFLG .EQ. 3 .AND. MINT .EQ. 1) THEN
            IF (BND.NE.0.E0) RH = MIN(RH, 1.E0/(2.E0*EL(1,NQ)*BND*ABS(H)))
          END IF
          CALL SDSCL (HMAX, N, NQ, RMAX,  H, RC, RH, YH)
          RMAX = RMNORM
 380      NWAIT = NQ + 2
        END IF
C                All returns are made through this section.  H is saved
C                in HOLD to allow the caller to change H on the next step
        JSTATE = 1
        HOLD = H
        RETURN
C
 400    JSTATE = 2
        HOLD = H
        DO 405 I = 1,N
 405      Y(I) = YH(I,1)
        RETURN
C
 410    JSTATE = 3
        HOLD = H
        RETURN
C
 420    JSTATE = 4
        HOLD = H
        RETURN
C
 430    T = TOLD
        CALL SDPSC (-1, NSV, NQ,  YH)
        DO 435 I = 1,NSV
 435      Y(I) = YH(I,1)
 440    HOLD = H
        RETURN
        END
        SUBROUTINE SDZRO (AE,F,H,N,NQ,IROOT,RE,T,YH,UROUND,B,C,FB,FC,Y)
C***BEGIN PROLOGUE  SDZRO
C***REFER TO  SDRIV3
C     This is a special purpose version of ZEROIN, modified for use with
C     the SDRIV1 package.
C
C     Sandia Mathematical Program Library
C     Mathematical Computing Services Division 5422
C     Sandia Laboratories
C     P. O. Box 5800
C     Albuquerque, New Mexico  87115
C     Control Data 6600 Version 4.5, 1 November 1971
C
C     ABSTRACT
```

```
C       ZEROIN searches for a zero of a function F(N, T, Y, IROOT)
C       between the given values B and C until the width of the
C       interval (B, C) has collapsed to within a tolerance specified
C       by the stopping criterion, ABS(B - C) .LE. 2.*(RW*ABS(B) + AE).
C
C     REFERENCES
C       1.  L F Shampine and H A Watts, ZEROIN, A Root-Solving Routine,
C           SC-TM-70-631, Sept 1970.
C       2.  T J Dekker, Finding a Zero by Means of Successive Linear
C           Interpolation, "Constructive Aspects of the Fundamental
C           Theorem of Algebra", edited by B Dejon and P Henrici, 1969.
C***ROUTINES CALLED  SDNTP
C***DATE WRITTEN   790601   (YYMMDD)
C***REVISION DATE  870511   (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C           SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDZRO
      REAL A, ACBS, ACMB, AE, B, C, CMB, ER, F, FA, FB, FC,
     8     H, P, Q, RE, RW, T, TOL, UROUND, Y(*), YH(N,*)
C***FIRST EXECUTABLE STATEMENT  SDZRO
      ER = 4.E0*UROUND
      RW = MAX(RE, ER)
      IC = 0
      ACBS = ABS(B - C)
      A = C
      FA = FC
      KOUNT = 0
C                                                  Perform interchange
 10   IF (ABS(FC) .LT. ABS(FB)) THEN
        A = B
        FA = FB
        B = C
        FB = FC
        C = A
        FC = FA
      END IF
      CMB = 0.5E0*(C - B)
      ACMB = ABS(CMB)
      TOL = RW*ABS(B) + AE
C                                               Test stopping criterion
      IF (ACMB .LE. TOL) RETURN
      IF (KOUNT .GT. 50) RETURN
C                                     Calculate new iterate implicitly as
C                                     B + P/Q, where we arrange P .GE. 0.
C                             The implicit form is used to prevent overflow.
      P = (B - A)*FB
      Q = FA - FB
      IF (P .LT. 0.E0) THEN
        P = -P
        Q = -Q
      END IF
C                             Update A and check for satisfactory reduction
C                             in the size of our bounding interval.
      A = B
      FA = FB
      IC = IC + 1
      IF (IC .GE. 4) THEN
        IF (8.E0*ACMB .GE. ACBS) THEN
C                                                                 Bisect
          B = 0.5E0*(C + B)
          GO TO 20
        END IF
        IC = 0
      END IF
      ACBS = ACMB
C                                           Test for too small a change
      IF (P .LE. ABS(Q)*TOL) THEN
C                                              Increment by tolerance
        B = B + SIGN(TOL, CMB)
C                                             Root ought to be between
C                                             B and (C + B)/2.
      ELSE IF (P .LT. CMB*Q) THEN
C                                                          Interpolate
        B = B + P/Q
      ELSE
C                                                               Bisect
        B = 0.5E0*(C + B)
      END IF
C                                            Have completed computation
C                                            for new iterate B.
 20   CALL SDNTP (H, 0, N, NQ, T, B, YH,  Y)
      FB = F(N, B, Y, IROOT)
      IF (N .EQ. 0) RETURN
      IF (FB .EQ. 0.E0) RETURN
      KOUNT = KOUNT + 1
C
C             Decide whether next step is interpolation or extrapolation
C
      IF (SIGN(1.0E0, FB) .EQ. SIGN(1.0E0, FC)) THEN
        C = A
        FC = FA
      END IF
      GO TO 10
      END
      SUBROUTINE SGBSL(ABD,LDA,N,ML,MU,IPVT,B,JOB)
C***BEGIN PROLOGUE  SGBSL
C***DATE WRITTEN   780814   (YYMMDD)
C***REVISION DATE  820801   (YYMMDD)
C***CATEGORY NO.  D2A2
C***KEYWORDS  BANDED,LINEAR ALGEBRA,LINPACK,MATRIX,SOLVE
C***AUTHOR  MOLER, C. B., (U. OF NEW MEXICO)
C***PURPOSE  Solves the real BAND system A*X=B or TRANS(A)*X=B
C            using the factors computed by SGBCO or SGBFA.
C***DESCRIPTION
C
C     SGBSL solves the real band system
C     A * X = B  or  TRANS(A) * X = B
C     using the factors computed by SBGCO or SGBFA.
C
C***REFERENCES  DONGARRA J.J., BUNCH J.R., MOLER C.B., STEWART G.W.,
C                 *LINPACK USERS  GUIDE*, SIAM, 1979.
```

```
C***ROUTINES CALLED  SAXPY,SDOT
C***END PROLOGUE  SGBSL
      INTEGER LDA,N,ML,MU,IPVT(1),JOB
      REAL ABD(LDA,1),B(1)
C
      REAL SDOT,T
      INTEGER K,KB,L,LA,LB,LM,M,NM1
C***FIRST EXECUTABLE STATEMENT  SGBSL
      M = MU + ML + 1
      NM1 = N - 1
      IF (JOB .NE. 0) GO TO 50
C
C        JOB = 0 , SOLVE  A * X = B
C        FIRST SOLVE L*Y = B
C
         IF (ML .EQ. 0) GO TO 30
         IF (NM1 .LT. 1) GO TO 30
            DO 20 K = 1, NM1
               LM = MIN0(ML,N-K)
               L = IPVT(K)
               T = B(L)
               IF (L .EQ. K) GO TO 10
                  B(L) = B(K)
                  B(K) = T
   10          CONTINUE
               CALL SAXPY(LM,T,ABD(M+1,K),1,B(K+1),1)
   20       CONTINUE
   30    CONTINUE
C
C        NOW SOLVE  U*X = Y
C
         DO 40 KB = 1, N
            K = N + 1 - KB
            B(K) = B(K)/ABD(M,K)
            LM = MIN0(K,M) - 1
            LA = M - LM
            LB = K - LM
            T = -B(K)
            CALL SAXPY(LM,T,ABD(LA,K),1,B(LB),1)
   40    CONTINUE
      GO TO 100
   50 CONTINUE
C
C        JOB = NONZERO, SOLVE  TRANS(A) * X = B
C        FIRST SOLVE  TRANS(U)*Y = B
C
         DO 60 K = 1, N
            LM = MIN0(K,M) - 1
            LA = M - LM
            LB = K - LM
            T = SDOT(LM,ABD(LA,K),1,B(LB),1)
            B(K) = (B(K) - T)/ABD(M,K)
   60    CONTINUE
C
C        NOW SOLVE TRANS(L)*X = Y
C
```

```
      IF (ML .EQ. 0) GO TO 90
      IF (NM1 .LT. 1) GO TO 90
         DO 80 KB = 1, NM1
            K = N - KB
            LM = MIN0(ML,N-K)
            B(K) = B(K) + SDOT(LM,ABD(M+1,K),1,B(K+1),1)
            L = IPVT(K)
            IF (L .EQ. K) GO TO 70
               T = B(L)
               B(L) = B(K)
               B(K) = T
   70       CONTINUE
   80    CONTINUE
   90 CONTINUE
  100 CONTINUE
      RETURN
      END
      SUBROUTINE SGEFA(A,LDA,N,IPVT,INFO)
C***BEGIN PROLOGUE  SGEFA
C***DATE WRITTEN   780814   (YYMMDD)
C***REVISION DATE  820801   (YYMMDD)
C***CATEGORY NO.  D2A1
C***KEYWORDS  FACTOR,LINEAR ALGEBRA,LINPACK,MATRIX
C***AUTHOR  MOLER, C. B., (U. OF NEW MEXICO)
C***PURPOSE  Factors a real matrix by Gaussian elimination.
C***DESCRIPTION
C
C     SGEFA factors a real matrix by Gaussian elimination.
C
C     SGEFA is usually called by SGECO, but it can be called
C     directly with a saving in time if  RCOND  is not needed.
C     (Time for SGECO) = (1 + 9/N)*(Time for SGEFA) .
C     LINPACK.  This version dated 08/14/78 .
C     Cleve Moler, University of New Mexico, Argonne National Lab.
C
C     Subroutines and Functions
C
C     BLAS SAXPY,SSCAL,ISAMAX
C***REFERENCES  DONGARRA J.J., BUNCH J.R., MOLER C.B., STEWART G.W.,
C                 *LINPACK USERS  GUIDE*, SIAM, 1979.
C***ROUTINES CALLED  ISAMAX,SAXPY,SSCAL
C***END PROLOGUE  SGEFA
      INTEGER LDA,N,IPVT(1),INFO
      REAL A(LDA,1)
C
      REAL T
      INTEGER ISAMAX,J,K,KP1,L,NM1
C
C     GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
C
C***FIRST EXECUTABLE STATEMENT  SGEFA
      INFO = 0
      NM1 = N - 1
      IF (NM1 .LT. 1) GO TO 70
      DO 60 K = 1, NM1
         KP1 = K + 1
```

```
C
C        FIND L = PIVOT INDEX
C
         L = ISAMAX(N-K+1,A(K,K),1) + K - 1
         IPVT(K) = L
C
C        ZERO PIVOT IMPLIES THIS COLUMN ALREADY TRIANGULARIZED
C
         IF (A(L,K) .EQ. 0.0E0) GO TO 40
C
C           INTERCHANGE IF NECESSARY
C
            IF (L .EQ. K) GO TO 10
               T = A(L,K)
               A(L,K) = A(K,K)
               A(K,K) = T
   10       CONTINUE
C
C           COMPUTE MULTIPLIERS
C
            T = -1.0E0/A(K,K)
            CALL SSCAL(N-K,T,A(K+1,K),1)
C
C           ROW ELIMINATION WITH COLUMN INDEXING
C
            DO 30 J = KP1, N
               T = A(L,J)
               IF (L .EQ. K) GO TO 20
                  A(L,J) = A(K,J)
                  A(K,J) = T
   20          CONTINUE
               CALL SAXPY(N-K,T,A(K+1,K),1,A(K+1,J),1)
   30       CONTINUE
         GO TO 50
   40    CONTINUE
            INFO = K
   50    CONTINUE
   60 CONTINUE
   70 CONTINUE
      IPVT(N) = N
      IF (A(N,N) .EQ. 0.0E0) INFO = N
      RETURN
      END
      SUBROUTINE SGESL(A,LDA,N,IPVT,B,JOB)
C***BEGIN PROLOGUE  SGESL
C***DATE WRITTEN   780814   (YYMMDD)
C***REVISION DATE  820801   (YYMMDD)
C***CATEGORY NO.  D2A1
C***KEYWORDS  LINEAR ALGEBRA,LINPACK,MATRIX,SOLVE
C***AUTHOR  MOLER, C. B., (U. OF NEW MEXICO)
C***PURPOSE  Solves the real system A*X=B or TRANS(A)*X=B
C            using the factors of SGECO or SGEFA
C***DESCRIPTION
C
C     SGESL solves the real system
C     A * X = B  or  TRANS(A) * X = B
```

```
C     using the factors computed by SGECO or SGEFA.
C
C***REFERENCES  DONGARRA J.J., BUNCH J.R., MOLER C.B., STEWART G.W.,
C                 *LINPACK USERS  GUIDE*, SIAM, 1979.
C***ROUTINES CALLED  SAXPY,SDOT
C***END PROLOGUE  SGESL
      INTEGER LDA,N,IPVT(1),JOB
      REAL A(LDA,1),B(1)
C
      REAL SDOT,T
      INTEGER K,KB,L,NM1
C***FIRST EXECUTABLE STATEMENT  SGESL
      NM1 = N - 1
      IF (JOB .NE. 0) GO TO 50
C
C        JOB = 0 , SOLVE  A * X = B
C        FIRST SOLVE  L*Y = B
C
         IF (NM1 .LT. 1) GO TO 30
         DO 20 K = 1, NM1
            L = IPVT(K)
            T = B(L)
            IF (L .EQ. K) GO TO 10
               B(L) = B(K)
               B(K) = T
   10       CONTINUE
            CALL SAXPY(N-K,T,A(K+1,K),1,B(K+1),1)
   20    CONTINUE
   30    CONTINUE
C
C        NOW SOLVE  U*X = Y
C
         DO 40 KB = 1, N
            K = N + 1 - KB
            B(K) = B(K)/A(K,K)
            T = -B(K)
            CALL SAXPY(K-1,T,A(1,K),1,B(1),1)
   40    CONTINUE
      GO TO 100
   50 CONTINUE
C
C        JOB = NONZERO, SOLVE  TRANS(A) * X = B
C        FIRST SOLVE  TRANS(U)*Y = B
C
         DO 60 K = 1, N
            T = SDOT(K-1,A(1,K),1,B(1),1)
            B(K) = (B(K) - T)/A(K,K)
   60    CONTINUE
C
C        NOW SOLVE TRANS(L)*X = Y
C
         IF (NM1 .LT. 1) GO TO 90
         DO 80 KB = 1, NM1
            K = N - KB
            B(K) = B(K) + SDOT(N-K,A(K+1,K),1,B(K+1),1)
            L = IPVT(K)
            IF (L .EQ. K) GO TO 70
               T = B(L)
               B(L) = B(K)
               B(K) = T
   70       CONTINUE
   80    CONTINUE
   90    CONTINUE
  100 CONTINUE
      RETURN
      END
      REAL FUNCTION SNRM2 ( N, SX, INCX)
      INTEGER          NEXT
      REAL   SX(1),  CUTLO, CUTHI, HITEST, SUM, XMAX, ZERO, ONE
      DATA   ZERO, ONE /0.0E0, 1.0E0/
C
C     EUCLIDEAN NORM OF THE N-VECTOR STORED IN SX() WITH STORAGE
C     INCREMENT INCX .
C     IF    N .LE. 0 RETURN WITH RESULT = 0.
C     IF N .GE. 1 THEN INCX MUST BE .GE. 1
C
C           C.L.LAWSON, 1978 JAN 08
C
C     FOUR PHASE METHOD     USING TWO BUILT-IN CONSTANTS THAT ARE
C     HOPEFULLY APPLICABLE TO ALL MACHINES.
C         CUTLO = MAXIMUM OF  SQRT(U/EPS)  OVER ALL KNOWN MACHINES.
C         CUTHI = MINIMUM OF  SQRT(V)      OVER ALL KNOWN MACHINES.
C     WHERE
C         EPS = SMALLEST NO. SUCH THAT EPS + 1. .GT. 1.
C         U   = SMALLEST POSITIVE NO.   (UNDERFLOW LIMIT)
C         V   = LARGEST  NO.            (OVERFLOW  LIMIT)
C
      DATA CUTLO, CUTHI / 4.441E-16,  1.304E19 /
C
      IF(N .GT. 0) GO TO 10
         SNRM2  = ZERO
         GO TO 300
C
   10 ASSIGN 30 TO NEXT
      SUM = ZERO
      NN = N * INCX
C                                                 BEGIN MAIN LOOP
      I = 1
   20    GO TO NEXT,(30, 50, 70, 110)
   30 IF( ABS(SX(I)) .GT. CUTLO) GO TO 85
      ASSIGN 50 TO NEXT
      XMAX = ZERO
C
C                        PHASE 1.  SUM IS ZERO
C
   50 IF( SX(I) .EQ. ZERO) GO TO 200
      IF( ABS(SX(I)) .GT. CUTLO) GO TO 85
C
C                                PREPARE FOR PHASE 2.
      ASSIGN 70 TO NEXT
      GO TO 105
C
```

```
C                                   PREPARE FOR PHASE 4.
C
  100 I = J
      ASSIGN 110 TO NEXT
      SUM = (SUM / SX(I)) / SX(I)
  105 XMAX = ABS(SX(I))
      GO TO 115
C
C                     PHASE 2.  SUM IS SMALL.
C                               SCALE TO AVOID DESTRUCTIVE UNDERFLOW.
C
   70 IF( ABS(SX(I)) .GT. CUTLO ) GO TO 75
C
C                      COMMON CODE FOR PHASES 2 AND 4.
C                      IN PHASE 4 SUM IS LARGE.  SCALE TO AVOID OVERFLOW.
C
  110 IF( ABS(SX(I)) .LE. XMAX ) GO TO 115
         SUM = ONE + SUM * (XMAX / SX(I))**2
         XMAX = ABS(SX(I))
         GO TO 200
C
  115 SUM = SUM + (SX(I)/XMAX)**2
      GO TO 200
C
C
C                     PREPARE FOR PHASE 3.
C
   75 SUM = (SUM * XMAX) * XMAX
C
C
C     FOR REAL OR D.P. SET HITEST = CUTHI/N
C     FOR COMPLEX      SET HITEST = CUTHI/(2*N)
C
   85 HITEST = CUTHI/FLOAT( N )
C
C                     PHASE 3.  SUM IS MID-RANGE.  NO SCALING.
C
      DO 95 J =I,NN,INCX
      IF(ABS(SX(J)) .GE. HITEST) GO TO 100
   95    SUM = SUM + SX(J)**2
      SNRM2 = SQRT( SUM )
      GO TO 300
C
  200 CONTINUE
      I = I + INCX
      IF ( I .LE. NN ) GO TO 20
C
C                END OF MAIN LOOP.
C
C                COMPUTE SQUARE ROOT AND ADJUST FOR SCALING.
C
      SNRM2 = XMAX * SQRT(SUM)
  300 CONTINUE
      RETURN
      END
      SUBROUTINE SSCAL(N,SA,SX,INCX)
```

```
C
C     REPLACE SINGLE PRECISION SX BY SINGLE PRECISION SA*SX.
C     FOR I = 0 TO N-1, REPLACE SX(1+I*INCX) WITH  SA * SX(1+I*INCX)
C
      REAL SA,SX(1)
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1)GOTO 20
C
C        CODE FOR INCREMENTS NOT EQUAL TO 1.
C
      NS = N*INCX
          DO 10 I = 1,NS,INCX
          SX(I) = SA*SX(I)
   10     CONTINUE
      RETURN
C
C        CODE FOR INCREMENTS EQUAL TO 1.
C
C
C        CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 5.
C
   20 M = MOD(N,5)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        SX(I) = SA*SX(I)
   30 CONTINUE
      IF( N .LT. 5 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,5
        SX(I) = SA*SX(I)
        SX(I + 1) = SA*SX(I + 1)
        SX(I + 2) = SA*SX(I + 2)
        SX(I + 3) = SA*SX(I + 3)
        SX(I + 4) = SA*SX(I + 4)
   50 CONTINUE
      RETURN
      END
      SUBROUTINE XERABT(MESSG,NMESSG)
C***BEGIN PROLOGUE  XERABT
C***DATE WRITTEN   790801   (YYMMDD)
C***REVISION DATE  820801   (YYMMDD)
C***CATEGORY NO.  R3C
C***KEYWORDS  ERROR,XERROR PACKAGE
C***AUTHOR  JONES, R. E., (SNLA)
C***PURPOSE  Aborts program execution and prints error message.
C***DESCRIPTION
C     Abstract
C        ***Note*** machine dependent routine
C        XERABT aborts the execution of the program.
C        The error message causing the abort is given in the calling
C        sequence, in case one needs it for printing on a dayfile,
C        for example.
C
C     Description of Parameters
C        MESSG and NMESSG are as in XERROR, except that NMESSG may
C        be zero, in which case no message is being supplied.
```

```
C
C     Written by Ron Jones, with SLATEC Common Math Library Subcommittee
C     Latest revision ---  19 MAR 1980
C***REFERENCES  JONES R.E., KAHANER D.K., "XERROR, THE SLATEC ERROR-
C                   HANDLING PACKAGE", SAND82-0800, SANDIA LABORATORIES,
C                   1982.
C***ROUTINES CALLED  (NONE)
C***END PROLOGUE  XERABT
      CHARACTER*(*) MESSG
C***FIRST EXECUTABLE STATEMENT  XERABT
      STOP
      END
      SUBROUTINE XERCLR
C***BEGIN PROLOGUE  XERCLR
C***DATE WRITTEN   790801   (YYMMDD)
C***REVISION DATE  820801   (YYMMDD)
C***CATEGORY NO.  R3C
C***KEYWORDS  ERROR,XERROR PACKAGE
C***AUTHOR  JONES, R. E., (SNLA)
C***PURPOSE  Resets current error number to zero.
C***DESCRIPTION
C     Abstract
C        This routine simply resets the current error number to zero.
C        This may be necessary to do in order to determine that
C        a certain error has occurred again since the last time
C        NUMXER was referenced.
C
C     Written by Ron Jones, with SLATEC Common Math Library Subcommittee
C     Latest revision ---  7 June 1978
C***REFERENCES  JONES R.E., KAHANER D.K., "XERROR, THE SLATEC ERROR-
C                   HANDLING PACKAGE", SAND82-0800, SANDIA LABORATORIES,
C                   1982.
C***ROUTINES CALLED  J4SAVE
C***END PROLOGUE  XERCLR
C***FIRST EXECUTABLE STATEMENT  XERCLR
      JUNK = J4SAVE(1,0,.TRUE.)
      RETURN
      END
      SUBROUTINE XERCTL(MESSG1,NMESSG,NERR,LEVEL,KONTRL)
C***BEGIN PROLOGUE  XERCTL
C***DATE WRITTEN   790801   (YYMMDD)
C***REVISION DATE  820801   (YYMMDD)
C***CATEGORY NO.  R3C
C***KEYWORDS  ERROR,XERROR PACKAGE
C***AUTHOR  JONES, R. E., (SNLA)
C***PURPOSE  Allows user control over handling of individual errors.
C***DESCRIPTION
C     Abstract
C        Allows user control over handling of individual errors.
C        Just after each message is recorded, but before it is
C        processed any further (i.e., before it is printed or
C        a decision to abort is made), a call is made to XERCTL.
C        If the user has provided his own version of XERCTL, he
C        can then override the value of KONTROL used in processing
C        this message by redefining its value.
C        KONTRL may be set to any value from -2 to 2.
```

```
C        The meanings for KONTRL are the same as in XSETF, except
C        that the value of KONTRL changes only for this message.
C        If KONTRL is set to a value outside the range from -2 to 2,
C        it will be moved back into that range.
C
C***REFERENCES  JONES R.E., KAHANER D.K., "XERROR, THE SLATEC ERROR-
C                 HANDLING PACKAGE", SAND82-0800, SANDIA LABORATORIES,
C                 1982.
C***ROUTINES CALLED  (NONE)
C***END PROLOGUE  XERCTL
      CHARACTER*20 MESSG1
C***FIRST EXECUTABLE STATEMENT  XERCTL
      RETURN
      END
      SUBROUTINE XERDMP
C***BEGIN PROLOGUE  XERDMP
C***DATE WRITTEN   790801   (YYMMDD)
C***REVISION DATE  820801   (YYMMDD)
C***CATEGORY NO.  R3C
C***KEYWORDS  ERROR,XERROR PACKAGE
C***AUTHOR  JONES, R. E., (SNLA)
C***PURPOSE  Prints the error tables and then clears them.
C***DESCRIPTION
C     Abstract
C        XERDMP prints the error tables, then clears them.
C
C     Written by Ron Jones, with SLATEC Common Math Library Subcommittee
C     Latest revision ---  7 June 1978
C***REFERENCES  JONES R.E., KAHANER D.K., "XERROR, THE SLATEC ERROR-
C                 HANDLING PACKAGE", SAND82-0800, SANDIA LABORATORIES,
C                 1982.
C***ROUTINES CALLED  XERSAV
C***END PROLOGUE  XERDMP
C***FIRST EXECUTABLE STATEMENT  XERDMP
      CALL XERSAV(' ',0,0,0,KOUNT)
      RETURN
      END
      SUBROUTINE XERMAX(MAX)
C***BEGIN PROLOGUE  XERMAX
C***DATE WRITTEN   790801   (YYMMDD)
C***REVISION DATE  820801   (YYMMDD)
C***CATEGORY NO.  R3C
C***KEYWORDS  ERROR,XERROR PACKAGE
C***AUTHOR  JONES, R. E., (SNLA)
C***PURPOSE  Sets maximum number of times any error message is to be
C            printed.
C***DESCRIPTION
C     Abstract
C        XERMAX sets the maximum number of times any message
C        is to be printed.  That is, non-fatal messages are
C        not to be printed after they have occured MAX times.
C        Such non-fatal messages may be printed less than
C        MAX times even if they occur MAX times, if error
C        suppression mode (KONTRL=0) is ever in effect.
C
C     Written by Ron Jones, with SLATEC Common Math Library Subcommittee
```

```
C     Latest revision ---  7 June 1978
C***REFERENCES  JONES R.E., KAHANER D.K., "XERROR, THE SLATEC ERROR-
C                 HANDLING PACKAGE", SAND82-0800, SANDIA LABORATORIES,
C                 1982.
C***ROUTINES CALLED  J4SAVE
C***END PROLOGUE  XERMAX
C***FIRST EXECUTABLE STATEMENT  XERMAX
      JUNK = J4SAVE(4,MAX,.TRUE.)
      RETURN
      END
      SUBROUTINE XERPRT(MESSG,NMESSG)
C***BEGIN PROLOGUE  XERPRT
C***DATE WRITTEN   790801   (YYMMDD)
C***REVISION DATE  820801   (YYMMDD)
C***CATEGORY NO.  Z
C***KEYWORDS  ERROR,XERROR PACKAGE
C***AUTHOR  JONES, R. E., (SNLA)
C***PURPOSE  Prints error messages.
C***DESCRIPTION
C     Abstract
C        Print the Hollerith message in MESSG, of length NMESSG,
C        on each file indicated by XGETUA.
C     Latest revision ---  19 MAR 1980
C***REFERENCES  JONES R.E., KAHANER D.K., "XERROR, THE SLATEC ERROR-
C                 HANDLING PACKAGE", SAND82-0800, SANDIA LABORATORIES,
C                 1982.
C***ROUTINES CALLED  I1MACH,S88FMT,XGETUA
C***END PROLOGUE  XERPRT
      INTEGER LUN(5)
      CHARACTER*(*) MESSC
C     OBTAIN UNIT NUMBERS AND WRITE LINE TO EACH UNIT
C***FIRST EXECUTABLE STATEMENT  XERPRT
      CALL XGETUA(LUN,NUNIT)
      LENMES = LEN(MESSG)
      DO 20 KUNIT=1,NUNIT
         IUNIT = LUN(KUNIT)
         IF (IUNIT.EQ.0) IUNIT = I1MACH(4)
         DO 10 ICHAR=1,LENMES,72
            LAST = MIN0(ICHAR+71 , LENMES)
            WRITE (IUNIT,'(1X,A)') MESSG(ICHAR:LAST)
   10    CONTINUE
   20 CONTINUE
      RETURN
      END
      SUBROUTINE XERROR(MESSG,NMESSG,NERR,LEVEL)
C***BEGIN PROLOGUE  XERROR
C***DATE WRITTEN   790801   (YYMMDD)
C***REVISION DATE  820801   (YYMMDD)
C***CATEGORY NO.  R3C
C***KEYWORDS  ERROR,XERROR PACKAGE
C***AUTHOR  JONES, R. E., (SNLA)
C***PURPOSE  Processes an error (diagnostic) message.
C***DESCRIPTION
C     Abstract
C        XERROR processes a diagnostic message, in a manner
C        determined by the value of LEVEL and the current value
```

```
C        of the library error control flag, KONTRL.
C        (See subroutine XSETF for details.)
C
C     Latest revision ---  19 MAR 1980
C     Written by Ron Jones, with SLATEC Common Math Library Subcommittee
C***REFERENCES  JONES R.E., KAHANER D.K., "XERROR, THE SLATEC ERROR-
C                 HANDLING PACKAGE", SAND82-0800, SANDIA LABORATORIES,
C                 1982.
C***ROUTINES CALLED  XERRWV
C***END PROLOGUE  XERROR
      CHARACTER*(*) MESSG
C***FIRST EXECUTABLE STATEMENT  XERROR
      CALL XERRWV(MESSG,NMESSG,NERR,LEVEL,0,0,0,0,0.,0.)
      RETURN
      END
      SUBROUTINE XERRWV(MESSG,NMESSG,NERR,LEVEL,NI,I1,I2,NR,R1,R2)
C***BEGIN PROLOGUE  XERRWV
C***DATE WRITTEN   800319   (YYMMDD)
C***REVISION DATE  820801   (YYMMDD)
C***CATEGORY NO.  R3C
C***KEYWORDS  ERROR,XERROR PACKAGE
C***AUTHOR  JONES, R. E., (SNLA)
C***PURPOSE  Processes error message allowing 2 integer and two real
C            values to be included in the message.
C***DESCRIPTION
C     Abstract
C        XERRWV processes a diagnostic message, in a manner
C        determined by the value of LEVEL and the current value
C        of the library error control flag, KONTRL.
C        (See subroutine XSETF for details.)
C        In addition, up to two integer values and two real
C        values may be printed along with the message.
C
C     Latest revision ---  19 MAR 1980
C     Written by Ron Jones, with SLATEC Common Math Library Subcommittee
C***REFERENCES  JONES R.E., KAHANER D.K., "XERROR, THE SLATEC ERROR-
C                 HANDLING PACKAGE", SAND82-0800, SANDIA LABORATORIES,
C                 1982.
C***ROUTINES CALLED  FDUMP,I1MACH,J4SAVE,XERABT,XERCTL,XERPRT,XERSAV,
C                    XGETUA
C***END PROLOGUE  XERRWV
      CHARACTER*(*) MESSG
      CHARACTER*20 LFIRST
      CHARACTER*37 FORM
      DIMENSION LUN(5)
C     GET FLAGS
C***FIRST EXECUTABLE STATEMENT  XERRWV
      LKNTRL = J4SAVE(2,0,.FALSE.)
      MAXMES = J4SAVE(4,0,.FALSE.)
C     CHECK FOR VALID INPUT
      IF ((NMESSG.GT.0).AND.(NERR.NE.0).AND.
     1    (LEVEL.GE.(-1)).AND.(LEVEL.LE.2)) GO TO 10
         IF (LKNTRL.GT.0) CALL XERPRT('FATAL ERROR IN...',17)
         CALL XERPRT('XERROR -- INVALID INPUT',23)
         IF (LKNTRL.GT.0) CALL FDUMP
         IF (LKNTRL.GT.0) CALL XERPRT('JOB ABORT DUE TO FATAL ERROR.',
     1  29)
           IF (LKNTRL.GT.0) CALL XERSAV(' ',0,0,0,KDUMMY)
           CALL XERABT('XERROR -- INVALID INPUT',23)
           RETURN
   10 CONTINUE
C     RECORD MESSAGE
      JUNK = J4SAVE(1,NERR,.TRUE.)
      CALL XERSAV(MESSG,NMESSG,NERR,LEVEL,KOUNT)
C     LET USER OVERRIDE
      LFIRST = MESSG
      LMESSG = NMESSG
      LERR = NERR
      LLEVEL = LEVEL
      CALL XERCTL(LFIRST,LMESSG,LERR,LLEVEL,LKNTRL)
C     RESET TO ORIGINAL VALUES
      LMESSG = NMESSG
      LERR = NERR
      LLEVEL = LEVEL
      LKNTRL = MAX0(-2,MIN0(2,LKNTRL))
      MKNTRL = IABS(LKNTRL)
C     DECIDE WHETHER TO PRINT MESSAGE
      IF ((LLEVEL.LT.2).AND.(LKNTRL.EQ.0)) GO TO 100
      IF (((LLEVEL.EQ.(-1)).AND.(KOUNT.GT.MIN0(1,MAXMES)))
     1.OR.((LLEVEL.EQ.0)   .AND.(KOUNT.GT.MAXMES))
     2.OR.((LLEVEL.EQ.1)   .AND.(KOUNT.GT.MAXMES).AND.(MKNTRL.EQ.1))
     3.OR.((LLEVEL.EQ.2)   .AND.(KOUNT.GT.MAX0(1,MAXMES)))) GO TO 100
         IF (LKNTRL.LE.0) GO TO 20
            CALL XERPRT(' ',1)
C           INTRODUCTION
            IF (LLEVEL.EQ.(-1)) CALL XERPRT
     1('WARNING MESSAGE...THIS MESSAGE WILL ONLY BE PRINTED ONCE.',57)
            IF (LLEVEL.EQ.0) CALL XERPRT('WARNING IN...',13)
            IF (LLEVEL.EQ.1) CALL XERPRT
     1      ('RECOVERABLE ERROR IN...',23)
            IF (LLEVEL.EQ.2) CALL XERPRT('FATAL ERROR IN...',17)
   20    CONTINUE
C        MESSAGE
         CALL XERPRT(MESSG,LMESSG)
         CALL XGETUA(LUN,NUNIT)
         ISIZEI = LOG10(FLOAT(I1MACH(9))) + 1.0
         ISIZEF = LOG10(FLOAT(I1MACH(10))**I1MACH(11)) + 1.0
         DO 50 KUNIT=1,NUNIT
            IUNIT = LUN(KUNIT)
            IF (IUNIT.EQ.0) IUNIT = I1MACH(4)
            DO 22 I=1,MIN(NI,2)
               WRITE (FORM,21) I,ISIZEI
   21          FORMAT ('(11X,21HIN ABOVE MESSAGE, I',I1,'=,I',I2,')   ')
               IF (I.EQ.1) WRITE (IUNIT,FORM) I1
               IF (I.EQ.2) WRITE (IUNIT,FORM) I2
   22       CONTINUE
            DO 24 I=1,MIN(NR,2)
               WRITE (FORM,23) I,ISIZEF+10,ISIZEF
   23          FORMAT ('(11X,21HIN ABOVE MESSAGE, R',I1,'=,E',
     1         I2,'.',I2,')')
               IF (I.EQ.1) WRITE (IUNIT,FORM) R1
               IF (I.EQ.2) WRITE (IUNIT,FORM) R2
   24          CONTINUE
               IF (LKNTRL.LE.0) GO TO 40
C                 ERROR NUMBER
                  WRITE (IUNIT,30) LERR
   30             FORMAT (15H ERROR NUMBER =,I10)
   40          CONTINUE
   50       CONTINUE
C           TRACE-BACK
            IF (LKNTRL.GT.0) CALL FDUMP
  100 CONTINUE
      IFATAL = 0
      IF ((LLEVEL.EQ.2).OR.((LLEVEL.EQ.1).AND.(MKNTRL.EQ.2)))
     1IFATAL = 1
C     QUIT HERE IF MESSAGE IS NOT FATAL
      IF (IFATAL.LE.0) RETURN
      IF ((LKNTRL.LE.0).OR.(KOUNT.GT.MAX0(1,MAXMES))) GO TO 120
C        PRINT REASON FOR ABORT
         IF (LLEVEL.EQ.1) CALL XERPRT
     1   ('JOB ABORT DUE TO UNRECOVERED ERROR.',35)
         IF (LLEVEL.EQ.2) CALL XERPRT
     1   ('JOB ABORT DUE TO FATAL ERROR.',29)
C        PRINT ERROR SUMMARY
         CALL XERSAV(' ',-1,0,0,KDUMMY)
  120 CONTINUE
C     ABORT
      IF ((LLEVEL.EQ.2).AND.(KOUNT.GT.MAX0(1,MAXMES))) LMESSG = 0
      CALL XERABT(MESSG,LMESSG)
      RETURN
      END
      SUBROUTINE XERSAV(MESSG,NMESSG,NERR,LEVEL,ICOUNT)
C***BEGIN PROLOGUE  XERSAV
C***DATE WRITTEN   800319   (YYMMDD)
C***REVISION DATE  820801   (YYMMDD)
C***CATEGORY NO.  Z
C***KEYWORDS  ERROR,XERROR PACKAGE
C***AUTHOR  JONES, R. E., (SNLA)
C***PURPOSE  Records that an error occurred.
C***DESCRIPTION
C     Abstract
C        Record that this error occurred.
C
C     Written by Ron Jones, with SLATEC Common Math Library Subcommittee
C     Latest revision ---  19 Mar 1980
C***REFERENCES  JONES R.E., KAHANER D.K., "XERROR, THE SLATEC ERROR-
C                 HANDLING PACKAGE", SAND82-0800, SANDIA LABORATORIES,
C                 1982.
C***ROUTINES CALLED  I1MACH,S88FMT,XGETUA
C***END PROLOGUE  XERSAV
      INTEGER LUN(5)
      CHARACTER*(*) MESSG
      CHARACTER*20 MESTAB(10),MES
      DIMENSION NERTAB(10),LEVTAB(10),KOUNT(10)
      SAVE MESTAB,NERTAB,LEVTAB,KOUNT,KOUNTX
C     NEXT TWO DATA STATEMENTS ARE NECESSARY TO PROVIDE A BLANK
C     ERROR TABLE INITIALLY
      DATA KOUNT(1),KOUNT(2),KOUNT(3  KOUNT(4),KOUNT(5),
     1     KOUNT(6),KOUNT(7),KOUNT(8),KOUNT(9),KOUNT(10)
     2     /0,0,0,0,0,0,0,0,0,0/
      DATA KOUNTX/0/
C***FIRST EXECUTABLE STATEMENT  XERSAV
      IF (NMESSG.GT.0) GO TO 80
C     DUMP THE TABLE
         IF (KOUNT(1).EQ.0) RETURN
C        PRINT TO EACH UNIT
         CALL XGETUA(LUN,NUNIT)
         DO 60 KUNIT-1,NUNIT
            IUNIT - LUN(KUNIT)
            IF (IUNIT.EQ.0) IUNIT - I1MACH(4)
C           PRINT TABLE HEADER
            WRITE (IUNIT,10)
   10       FORMAT (32H0          ERROR MESSAGE SUMMARY/
     1      51H MESSAGE START             NERR     LEVEL     COUNT)
C           PRINT BODY OF TABLE
            DO 20 I-1,10
               IF (KOUNT(I).EQ.0) GO TO 30
               WRITE (IUNIT,15) MESTAB(I),NERTAB(I),LEVTAB(I),KOUNT(I)
   15          FORMAT (1X,A20,3I10)
   20       CONTINUE
   30       CONTINUE
C           PRINT NUMBER OF OTHER ERRORS
            IF (KOUNTX.NE.0) WRITE (IUNIT,40) KOUNTX
   40       FORMAT (41H0OTHER ERRORS NOT INDIVIDUALLY TABULATED-,I10)
            WRITE (IUNIT,50)
   50       FORMAT (1X)
   60    CONTINUE
         IF (NMESSG.LT.0) RETURN
C        CLEAR THE ERROR TABLES
         DO 70 I-1,10
   70       KOUNT(I) - 0
         KOUNTX - 0
         RETURN
   80 CONTINUE
C     PROCESS A MESSAGE...
C     SEARCH FOR THIS MESSG, OR ELSE AN EMPTY SLOT FOR THIS MESSG,
C     OR ELSE DETERMINE THAT THE ERROR TABLE IS FULL.
      MES - MESSG
      DO 90 I-1,10
         II - I
         IF (KOUNT(I).EQ.0) GO TO 110
         IF (MES.NE.MESTAB(I)) GO TO 90
         IF (NERR.NE.NERTAB(I)) GO TO 90
         IF (LEVEL.NE.LEVTAB(I)) GO TO 90
         GO TO 100
   90 CONTINUE
C     THREE POSSIBLE CASES...
C     TABLE IS FULL
         KOUNTX - KOUNTX+1
         ICOUNT - 1
         RETURN
C     MESSAGE FOUND IN TABLE
  100    KOUNT(II) - KOUNT(II) + 1
         ICOUNT - KOUNT(II)
         RETURN
C      EMPTY SLOT FOUND FOR NEW MESSAGE
  110    MESTAB(II) = MES
         NERTAB(II) = NERR
         LEVTAB(II) = LEVEL
         KOUNT(II)  = 1
         ICOUNT = 1
         RETURN
      END
      SUBROUTINE XGETF(KONTRL)
C***BEGIN PROLOGUE  XGETF
C***DATE WRITTEN   790801   (YYMMDD)
C***REVISION DATE  820801   (YYMMDD)
C***CATEGORY NO.  R3C
C***KEYWORDS  ERROR,XERROR PACKAGE
C***AUTHOR  JONES, R. E., (SNLA)
C***PURPOSE  Returns current value of error control flag.
C***DESCRIPTION
C   Abstract
C        XGETF returns the current value of the error control flag
C        in KONTRL.  See subroutine XSETF for flag value meanings.
C        (KONTRL is an output parameter only.)
C
C     Written by Ron Jones, with SLATEC Common Math Library Subcommittee
C     Latest revision ---  7 June 1978
C***REFERENCES  JONES R.E., KAHANER D.K., "XERROR, THE SLATEC ERROR-
C                 HANDLING PACKAGE", SAND82-0800, SANDIA LABORATORIES,
C                 1982.
C***ROUTINES CALLED  J4SAVE
C***END PROLOGUE  XGETF
C***FIRST EXECUTABLE STATEMENT  XGETF
      KONTRL = J4SAVE(2,0,.FALSE.)
      RETURN
      END
      SUBROUTINE XGETUA(IUNITA,N)
C***BEGIN PROLOGUE  XGETUA
C***DATE WRITTEN   790801   (YYMMDD)
C***REVISION DATE  820801   (YYMMDD)
C***CATEGORY NO.  R3C
C***KEYWORDS  ERROR,XERROR PACKAGE
C***AUTHOR  JONES, R. E., (SNLA)
C***PURPOSE  Returns unit number(s) to which error messages are being
C            sent.
C***DESCRIPTION
C     Abstract
C        XGETUA may be called to determine the unit number or numbers
C        to which error messages are being sent.
C        These unit numbers may have been set by a call to XSETUN,
C        or a call to XSETUA, or may be a default value.
C
C     Latest revision ---  19 MAR 1980
C     Written by Ron Jones, with SLATEC Common Math Library Subcommittee
C***REFERENCES  JONES R.E., KAHANER D.K., "XERROR, THE SLATEC ERROR-
C                 HANDLING PACKAGE", SAND82-0800, SANDIA LABORATORIES,
C                 1982.
C***ROUTINES CALLED  J4SAVE
```

```
C***END PROLOGUE  XGETUA
      DIMENSION IUNITA(5)
C***FIRST EXECUTABLE STATEMENT  XGETUA
      N = J4SAVE(5,0,.FALSE.)
      DO 30 I=1,N
         INDEX = I+4
         IF (I.EQ.1) INDEX = 3
         IUNITA(I) = J4SAVE(INDEX,0,.FALSE.)
   30 CONTINUE
      RETURN
      END
      SUBROUTINE XGETUN(IUNIT)
C***BEGIN PROLOGUE  XGETUN
C***DATE WRITTEN   790801   (YYMMDD)
C***REVISION DATE  820801   (YYMMDD)
C***CATEGORY NO.  R3C
C***KEYWORDS  ERROR,XERROR PACKAGE
C***AUTHOR  JONES, R. E., (SNLA)
C***PURPOSE  Returns the (first) output file to which messages are being
C            sent.
C***DESCRIPTION
C     Abstract
C        XGETUN gets the (first) output file to which error messages
C        are being sent.  To find out if more than one file is being
C        used, one must use the XGETUA routine.
C
C     Written by Ron Jones, with SLATEC Common Math Library Subcommittee
C     Latest revision --- 23 May 1979
C***REFERENCES  JONES R.E., KAHANER D.K., "XERROR, THE SLATEC ERROR-
C                 HANDLING PACKAGE", SAND82-0800, SANDIA LABORATORIES,
C                 1982.
C***ROUTINES CALLED  J4SAVE
C***END PROLOGUE  XGETUN
C***FIRST EXECUTABLE STATEMENT  XGETUN
      IUNIT = J4SAVE(3,0,.FALSE.)
      RETURN
      END
      SUBROUTINE XSETF(KONTRL)
C***BEGIN PROLOGUE  XSETF
C***DATE WRITTEN   790801   (YYMMDD)
C***REVISION DATE  820801   (YYMMDD)
C***CATEGORY NO.  R3A
C***KEYWORDS  ERROR.XERROR PACKAGE
C***AUTHOR  JONES, R. E., (SNLA)
C***PURPOSE  Sets the error control flag.
C***DESCRIPTION
C     Abstract
C        XSETF sets the error control flag value to KONTRL.
C        (KONTRL is an input parameter only.)
C        The following table shows how each message is treated,
C        depending on the values of KONTRL and LEVEL.  (See XERROR
C        for description of LEVEL.)
C
C        If KONTRL is zero or negative, no information other than the
C        message itself (including numeric values, if any) will be
C        printed.  If KONTRL is positive, introductory messages,
C        trace-backs, etc., will be printed in addition to the message.
C
C     Written by Ron Jones, with SLATEC Common Math Library Subcommittee
C     Latest revision ---  19 MAR 1980
C***REFERENCES  JONES R.E., KAHANER D.K., "XERROR, THE SLATEC ERROR-
C                  HANDLING PACKAGE", SAND82-0800, SANDIA LABORATORIES,
C                  1982.
C***ROUTINES CALLED  J4SAVE,XERRWV
C***END PROLOGUE  XSETF
C***FIRST EXECUTABLE STATEMENT  XSETF
      IF ((KONTRL.GE.(-2)).AND.(KONTRL.LE.2)) GO TO 10
         CALL XERRWV('XSETF  -- INVALID VALUE OF KONTRL (I1).',33,1,2,
     1  1,KONTRL,0,0,0.,0.)
         RETURN
   10 JUNK = J4SAVE(2,KONTRL,.TRUE.)
      RETURN
      END
      SUBROUTINE XSETUA(IUNITA,N)
C***BEGIN PROLOGUE  XSETUA
C***DATE WRITTEN   790801   (YYMMDD)
C***REVISION DATE  820801   (YYMMDD)
C***CATEGORY NO.  R3B
C***KEYWORDS  ERROR,XERROR PACKAGE
C***AUTHOR  JONES, R. E., (SNLA)
C***PURPOSE  Sets up to 5 unit numbers to which messages are to be sent.
C***DESCRIPTION
C     Abstract
C        XSETUA may be called to declare a list of up to five
C        logical units, each of which is to receive a copy of
C        each error message processed by this package.
C        The purpose of XSETUA is to allow simultaneous printing
C        of each error message on, say, a main output file,
C        an interactive terminal, and other files such as graphics
C        communication files.
C
C     Written by Ron Jones, with SLATEC Common Math Library Subcommittee
C     Latest revision ---  19 MAR 1980
C***REFERENCES  JONES R.E., KAHANER D.K., "XERROR, THE SLATEC ERROR-
C                  HANDLING PACKAGE", SAND82-0800, SANDIA LABORATORIES,
C                  1982.
C***ROUTINES CALLED  J4SAVE,XERRWV
C***END PROLOGUE  XSETUA
      DIMENSION IUNITA(5)
C***FIRST EXECUTABLE STATEMENT  XSETUA
      IF ((N.GE.1).AND.(N.LE.5)) GO TO 10
         CALL XERRWV('XSETUA -- INVALID VALUE OF N (I1).',34,1,2,
     1  1,N,0,0,0.,0.)
         RETURN
   10 CONTINUE
      DO 20 I=1,N
         INDEX = I+4
         IF (I.EQ.1) INDEX = 3
         JUNK = J4SAVE(INDEX,IUNITA(I),.TRUE.)
   20 CONTINUE
      JUNK = J4SAVE(5,N,.TRUE.)
      RETURN
      END
      SUBROUTINE XSETUN(IUNIT)
C***BEGIN PROLOGUE  XSETUN
C***DATE WRITTEN   790801   (YYMMDD)
C***REVISION DATE  820801   (YYMMDD)
C***CATEGORY NO.  R3B
C***KEYWORDS  ERROR,XERROR PACKAGE
C***AUTHOR  JONES, R. E., (SNLA)
C***PURPOSE  Sets output file to which error messages are to be sent.
C***DESCRIPTION
C     Abstract
C        XSETUN sets the output file to which error messages are to
C        be sent.  Only one file will be used.  See XSETUA for
C        how to declare more than one file.
C
C     Written by Ron Jones, with SLATEC Common Math Library Subcommittee
C     Latest revision ---  7 June 1978
C***REFERENCES  JONES R.E., KAHANER D.K., "XERROR, THE SLATEC ERROR-
C                 HANDLING PACKAGE", SAND82-0800, SANDIA LABORATORIES,
C                 1982.
C***ROUTINES CALLED  J4SAVE
C***END PROLOGUE  XSETUN
C***FIRST EXECUTABLE STATEMENT  XSETUN
      JUNK = J4SAVE(3,IUNIT,.TRUE.)
      JUNK = J4SAVE(5,1,.TRUE.)
      RETURN
      END
```

INTENTIONALLY LEFT BLANK.

No of Copies	Organization
(Unclass., unlimited) 12 (Unclass., limited) 2 (Classified) 2	Administrator Defense Technical Info Center ATTN: DTIC-DDA Cameron Station Alexandria, VA 22304-6145
1	HQDA (SARD-TR) WASH DC 20310-0001
1	Commander US Army Materiel Command ATTN: AMCDRA-ST 5001 Eisenhower Avenue Alexandria, VA 22333-0001
1	Commander US Army Laboratory Command ATTN: AMSLC-DL Adelphi, MD 20783-1145
2	Commander Armament RD&E Center US Army AMCCOM ATTN: SMCAR-MSI Picatinny Arsenal, NJ 07806-5000
2	Commander Armament RD&E Center US Army AMCCOM ATTN: SMCAR-TDC Picatinny Arsenal, NJ 07806-5000
1	Director Benet Weapons Laboratory Armament RD&E Center US Army AMCCOM ATTN: SMCAR-LCB-TL Watervliet, NY 12189-4050
1	Commander US Army Armament, Munitions and Chemical Command ATTN: SMCAR-ESP-L Rock Island, IL 61299-5000
1	Commander US Army Aviation Systems Command ATTN: AMSAV-DACL 4300 Goodfellow Blvd. St. Louis, MO 63120-1798
1	Director US Army Aviation Research and Technology Activity Ames Research Center Moffett Field, CA 94035-1099

No of Copies	Organization
1	Commander US Army Missile Command ATTN: AMSMI-RD-CS-R (DOC) Redstone Arsenal, AL 35898-5010
1	Commander US Army Tank Automotive Command ATTN: AMSTA-TSL (Technical Library) Warren, MI 48397-5000
1	Director US Army TRADOC Analysis Command ATTN: ATAA-SL White Sands Missile Range, NM 88002-5502
(Class. only) 1	Commandant US Army Infantry School ATTN: ATSH-CD (Security Mgr.) Fort Benning, GA 31905-5660
(Unclass. only) 1	Commandant US Army Infantry School ATTN: ATSH-CD-CSO-OR Fort Benning, GA 31905-5660
(Class. only) 1	The Rand Corporation P.O. Box 2138 Santa Monica, CA 90401-2138
1	Air Force Armament Laboratory ATTN: AFATL/DLODL Eglin AFB, FL 32542-5000

Aberdeen Proving Ground

Dir, USAMSAA
ATTN: AMXSY-D
AMXSY-MP, H. Cohen
Cdr, USATECOM
ATTN: AMSTE-TO-F
Cdr, CRDEC, AMCCOM
ATTN: SMCCR-RSP-A
SMCCR-MU
SMCCR-MSI
Dir, VLAMO
ATTN: AMSLC-VL-D

No. of Copies	Organization
2	Director Defense Advanced Research Projects Agency ATTN: J. Lupo J. Richardson 1400 Wilson Boulevard Arlington, VA 22209
2	HQDA ATTN: SARD-TR, B. Zimmerman I. Szkrybalo Washington, DC 20310
1	HQ, US Army Materiel Command ATTN: AMCICP-AD, B. Dunetz 5001 Eisenhower Avenue Alexandria, VA 22333-0001
13	Cmdr, Armament R&D Center US Army AMCCOM ATTN: SMCAR-TSS SMCAR-AEE-BR, B. Brodman SMCAR-AEE-B, D. Downs SMCAR-AEE-BR, W. Seals A. Beardell SMCAR-AEE-W, N. Slagg SMCAR-AEE, A. Bracuti J. Lannon M. Gupta J. Salo D. Chieu SMCAR-FSS-D, L. Frauen SMCAR-FSA-S, H. Liberman Picatinny Arsenal, NJ07806-5000
3	Commander Armament R&D Center US Army AMCCOM ATTN: SMCAR-FSS-DA, Bldg 94 J. Feneck R. Kopmann J. Irizarry Picatinny Arsenal, NJ 07806-5000

No. of Copies	Organization
4	Director Benet Weapons Laboratory Armament RD&E Center US Army AMCCOM ATTN: SMCAR-CCB-DS, E. Conroy A. Graham SMCAR-CCB, L. Johnson SMCAR-CCB-S, F. Heiser Watervliet, NY 12189-4050
1	Commander Materials Technology Lab US Army Laboratory Cmd ATTN: SLCMT-MCM-SB M. Levy Watertown, MA 02172-0001
1	Commander CECOM R&D Technical Library ATTN: ASNC-ELC-IT (Rpts Sec) Ft. Monmouth, NJ 07703-5301
1	Commander US Army Laboratory Cmd ATTN: SLCHD-TA-L 2800 Powder Mill Rd Adelphi, MD 20783-1145

DISTRIBUTION LIST

No. of Copies	Organization
1	Commander US Army Belvoir R&D Ctr ATTN: STRBE-WC Tech Library (Vault) B-315 Fort Belvoir, VA 22060-5606
1	Commander US Army Research Office ATTN: Tech Library PO Box 12211 Research Triangle Park, NC 27709-2211
1	Commander Armament Rsch & Dev Ctr US Army Armament, Munitions and Chemical Command ATTN: SMCAR-CCS-C, T Hung Picatinny Arsenal, NJ 07806-5000
2	Commandant US Army Field Artillery School ATTN: ATSF-CMW ATSF-TSM-CN, J. Spicer Ft Sill, OK 73503
1	Commandant US Army Armor Center ATTN: ATSB-CD-MLD Ft Knox, KY 40121
1	Commander Naval Surface Weapons Center ATTN: D.A. Wilson, Code G31 Dahlgren, VA 22448-5000
1	Commander Naval Surface Weapons Center ATTN: Code G33, J. East Dahlgren, VA 22448-5000
2	Commander US Naval Surface Weapons Ctr ATTN: O. Dengel K. Thorsted Silver Spring, MD 20902-5000
1	Commander Naval Weapons Center China Lake, CA 93555-6001
1	Superintendent Naval Postgraduate School Dept of Mechanical Engr ATTN: Code 1424, Library Monterey, CA 93943
1	AFOSR/NA (L. Caveny) Bldg 410 Bolling AFB, DC 20332
1	Commandant USAFAS ATTN: ATSF-TSM-CN Ft Sill, OK 73503-5600
1	Director Jet Propulsion Lab ATTN: Tech Library 4800 Oak Grove Drive Pasadena, CA 91109
2	Director National Aeronautics and Space Administration ATTN: MS-603, Tech Lib MS-86, Dr. Povinelli 21000 Brookpark Road Lewis Research Center Cleveland, OH 44135
1	Director National Aeronautics and Space Administration Manned Spacecraft Center Houston, TX 77058

No. of Copies	Organization
10	Central Intelligence Agency Office of Central Reference Dissemination Branch Room GE-47 HQS Washington, DC 20502
1	Central Intelligence Agency ATTN: Joseph E. Backofen HQ Room 5F22 Washington, DC 20505
1	Calspan Corporation ATTN: Tech Library PO Box 400 Buffalo, NY 14225
8	General Electric Ord Sys Div ATTN: J. Mandzy, OP43-220 R.E. Mayer H. West W. Pasko R. Pate I. Magoon J. Scudiere Minh Luu 100 Plastics Avenue Pittsfield, MA 01201-3698
1	General Electric Company Armament Systems Department ATTN: D. Maher Burlington, VT 05401
1	Honeywell Inc. ATTN: R.E. Tompkins MN38-3300 10400 Yellow Circle Drive Minnetonka, MN 55343
1	IITRI ATTN: Library 10 W. 35th St Chicago, IL 60616

No. of Copies	Organization
1	Olin Chemicals Research ATTN: David Gavin PO Box 586 Chesire, CT 06410-0586
2	Olin Corporation ATTN: Victor A. Corso Dr. Ronald L. Dotson PO Box 30-9644 New Haven, CT 06536
1	Paul Gough Associates ATTN: Paul Gough 1048 South Street Portsmouth, NH 03801-5423
1	Safety Consulting Engr ATTN: Mr. C. James Dahn 5240 Pearl St Rosemont, IL 60018
1	Sandia National Laboratories ATTN: R. Rychnovsky, Div 8152 PO Box 969 Livermore, CA 94551-0969
1	Sandia National Laboratories ATTN: S. Griffiths, Div 8244 PO Box 969 Livermore, CA 94551-0969
1	Sandia National Laboratories ATTN: R. Carling, Div 8152 PO Box 969 Livermore, CA 94551-0969
1	Science Applications, Inc. ATTN: R. Edelman 23146 Cumorah Crest Woodland Hills, CA 91364

No. of Copies	Organization
2	Science Applications Int'l Corporation ATTN: Dr. F. T. Phillips Dr. Fred Su 10210 Campus Point Drive San Diego, CA 92121
1	Science Applications Int'l Corporation ATTN: Norman Banks 4900 Waters Edge Drive Suite 255 Raleigh, NC 27606
1	Sundstrand Aviation Operations ATTN: Mr. Owen Briles PO Box 7202 Rockford, IL 61125
1	Veritay Technology, Inc. ATTN: E.B. Fisher 4845 Millersport Highway PO Box 305 East Amherst, NY 14051-0305
1	Director Applied Physics Laboratory The Johns Hopkins Univ. Johns Hopkins Road Laurel, MD 20707
2	Director CPIA The Johns Hopkins Univ. ATTN: T. Christian Tech Library Johns Hopkins Road Laurel, MD 20707
1	U. of Illinois at Chicago ATTN: Professor Sohail Murad Dept of Chemical Engr Box 4348 Chicago, IL 60680
1	U. of MD at College Park ATTN: Professor Franz Kasler Department of Chemistry College Park, MD 20742
1	U. of Missouri at Columbia ATTN: Professor R. Thompson Department of Chemistry Columbia, MO 65211
1	U. of Michigan ATTN: Prof. Gerard M. Faeth Dept of Aerospace Engr Ann Arbor, MI 48109-3796
1	U. of Missouri at Columbia ATTN: Professor F.K. Ross Research Reactor Columbia, MO 65211
1	U. of Missouri at Kansas City Department of Physics ATTN: Prof. R.D. Murphy 1110 East 48th Street Kansas City, MO 64110-2499
1	Pennsylvania State University Dept of Mechanical Engr ATTN: Prof. K. Kuo University Park, PA 16802
2	Princeton Combustion Rsch Laboratories, Inc. ATTN: N.A. Messina M. Summerfield 4275 US Highway One North Monmouth Junction, NJ 08852
1	University of Arkansas Dept of Chemical Engr ATTN: J. Havens 227 Engineering Building Fayetteville, AR 72701

No. of Copies	Organization
3	University of Delaware Department of Chemistry ATTN: Mr. James Cronin Professor Thomas Brill Mr. Peter Spohn Newark, DE 19711
1	U. of Texas at Austin Bureau of Engineering Rsch ATTN: BRC EME133, Room 1.100 H. Fair 10100 Burnet Road Austin, TX 78758

No. of Copies	Organization
1	Dr. Clive Woodley GS2 Division Building R31 RARDE Ft. Halstead Sevenoaks, Kent TN14 7BT England

INTENTIONALLY LEFT BLANK.

USER EVALUATION SHEET/CHANGE OF ADDRESS

This laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers below will aid us in our efforts.

1. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.) ______

2. How, specifically, is the report being used? (Information source, design data, procedure, source of ideas, etc.) ______

3. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided, or efficiencies achieved, etc? If so, please elaborate. ______

4. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.) ______

BRL Report Number ______ Division Symbol ______

Check here if desire to be removed from distribution list. ____

Check here for address change. ____

Current address: Organization ______
Address ______

--------------------------------FOLD AND TAPE CLOSED--------------------------

Director
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-5066

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE $300

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY LABEL
FIRST CLASS PERMIT NO. 12062 WASHINGTON D.C.

POSTAGE WILL BE PAID BY DEPARTMENT OF THE ARMY

Director
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-9989